UNIVERSITY OF ILLINOIS-URBANA
510.84 IL6R no. C002 no.916-920(1978
Continued-product algorithms for divisio
3 0112 088403727

Digitized by the Internet Archive
in 2013

http://archive.org/details/methodsforautoma916lars

570.84
Il6N
no. 916
cop. 2

Report No. UIUCDCS-R-78-916

UILU-ENG 78 1709

METHODS FOR AUTOMATIC ERROR ANALYSIS OF NUMERICAL ALGORITHMS

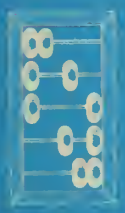
205

by

John Leonard Larson

April 1978

NSF-OCA-MCS75-21758-000033



The Library of the
JUN 15 1978
iversity of Illinois
bana-Cha

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. UIUCDCS-R-78-916

METHODS FOR AUTOMATIC ERROR ANALYSIS OF NUMERICAL ALGORITHMS*

by

John Leonard Larson

April 1978

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

*This work was supported in part by the National Science Foundation under Grant No. US NSF-MCS75-21758 and was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science, April 1978.

510.84
I26n
no. 916-925
cop. 2

METHODS FOR AUTOMATIC ERROR ANALYSIS OF NUMERICAL ALGORITHMS

John Leonard Larson, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1978

Using Bauer's approach to relative error propagation, methods are presented for performing various error analyses on numerical algorithms. A technique is developed which generates a system of equations employed by the error analyses, and requires an order of magnitude less time and storage than present techniques. Accurate analyses are attempted by working in the ∞-norm, and require the solution of minimax problems. A heuristic approximation method is also described. These methods are compared with the 2-norm approximation methods of Miller.

The error analyses provide alternative criteria by which algorithms that solve the same problem may be compared. Additionally, the analysis of a composite algorithm, which is made up of concatenated sub-algorithms is given in terms of analyses done on its parts. Finally, by using a forward analysis, limited algorithm improvement is obtained by the location of sensitive operations, and the substitution of mathematically equivalent expressions.

Acknowledgments

I would like to express my sincere appreciation to my thesis advisor, Professor Ahmed H. Sameh, for his guidance, insight, encouragement, and friendship. I would also like to thank the members of my thesis committee for their interest and comments: Professors David Kuck, Dave Liu, Paul Saylor, and Robert Skeel. Additional thanks go to Professor Webb Miller for providing his software routines and for his interest in my work.

I am grateful to the National Science Foundation and the Department of Computer Science for their financial support. Thanks are due to Ms. Cathy Gallion for an excellent job of typing.

Finally, I would like to thank my wife, Emmylou, for her love, patience and encouragement, and for her skill in preparing the illustrations.

Table of Contents

		Page
1.	Introduction	1
2.	Preliminaries	6
3.	Roundoff Errors	11
4.	Efficient Calculation of $\overline{M}$	19
5.	Error Analyses	25
	5.1. Forward Analysis	25
	5.2. Backward Analysis	27
	5.3. B-analysis	34
6.	Computational Details	39
	6.1. The Form of g*	39
	6.2. Simplification of Polytope Representation	40
	6.3. Algorithms for g*	48
7.	Comparison of Algorithms	66
8.	Composite Algorithms	77
9.	Algorithm Improvement	85
10.	Conclusion	94
	Bibliography	97
	Appendix	99
	Vita	102

1. Introduction

- New algorithms are constantly being proposed for the solution of numerical problems. In particular, these include algorithms which exploit the capabilities of new and emerging parallel and pipeline machines. Unfortunately, this enthusiasm has not been matched by the development of systematic ways to evaluate the numerical properties of these new algorithms. In fact, until recently [Miller, 8-12] little work has been done in the area of automating error analysis for existing sequential algorithms. Besides J. H. Wilkinson, few people have the time or inclination to make tedious error analyses of the algorithms they use. Error analysis needs to be organized in such a way that it is easily done by machine. This thesis is concerned with the development and implementation of methods for automatic error analysis of numerical algorithms.

This work parallels that of Miller [10], but takes a different approach to the analysis of numerical algorithms. His development is from an absolute error point of view, while this thesis is concerned with relative errors, a natural approach considering that floating point arithmetic is defined in terms of relative error. Perhaps the greatest difference in the approaches lies in their goals. While he strives to find data for which an approximate analysis indicates that the algorithm under investigation performs poorly, this thesis strives to give an accurate analysis of the algorithm for some given data. Each goal has its place depending on the need for the analysis.

Section 2 begins with a summary of previous work, and shows how roundoff errors affect computations. Aided by the representation of an algorithm as a directed graph, each numerical quantity is a node, and directed

arcs lead from operands to results. During the execution of the algorithm, the value at each node is subject to roundoff error, called local relative error as described in section 3. The local relative error in a node subsequently affects the value of other nodes. In particular, it affects those nodes which use the contaminated node as an operand. The magnitude of the effect on one of these nodes is reflected by a number which weights the arc between it and the contaminated node. Moreover, the local relative error in one node affects the value of another node if there exists a directed path between them. Additionally, the effect of the local relative error is the product of the arc weights along this path.

The effects of all local relative errors combine to produce a total relative error in each node. The definition of total relative error, when described in this way, is called the global definition, since all the nodes in the graph are inspected. An alternative formulation, called the local definition, recursively defines the total relative error in a node in terms of the total relative error in its operands. Each definition gives rise to a system of equations which relates the local and total relative errors. The system corresponding to the global definition is dense, while the local definition, as expected, produces an exceedingly sparse system, no more than four non-zero elements per row. The equivalence of these two systems is shown.

Either of these systems could be used to perform various error analyses. However, only those rows of the dense system corresponding to output nodes need be used. Current methods generate the entire dense system, and then discard all but the desired rows. Section 4 derives an algorithm which relies on the equivalence relation and computes only those

rows which are needed by solving sparse upper triangular systems. A comparison of this method with an existing alternative shows an order of magnitude savings in both time and storage.

Section 5 poses the problems to be solved in order to perform forward, backward, and B-analysis. A forward analysis determines relative error bounds for the computed solution. This bound may be easily computed from the system relating the local and total relative errors. A backward analysis determines the stability of an algorithm by finding a bound on relative input perturbations such that the computed solution satisfies the perturbed problem. This bound is found by comparing the set of output relative error vectors for all computed solutions with the set produced by local relative errors in only the input nodes. Conditions are specified for when a backward analysis is not possible. A B-analysis is an alternative stability measurement which determines a bound on relative input and output perturbations such that the computed solution is near the exact solution of a perturbed problem. This bound is found by comparing the set of output relative error vectors for all computed solutions with the set generated by local relative errors in only input and output nodes. An advantage of the B-analysis is that it may always be done.

Section 6 analyzes the problems posed in section 5. The problems for backward and B-analysis are identified as minimax problems, whose solutions are often tedious to obtain. These problems involve the expansion of a polytope (a centrally symmetric convex set) to cover another polytope. The polytopes represent the sets of output relative error vectors described above. Two exhaustive methods are described. The first method

searches the vertices of the polytope being covered for the vertex of maximal relative distance. The second method searches the faces of the polytope doing the covering. Each method makes use of the particular form of the solution. Additionally, a close inspection of the problems reveals that their complexity may often be significantly reduced. This is reflected in a reduction of the number of columns in the system which relates local and total relative errors. Nevertheless, the computation time of the exhaustive methods precludes their use for all but small problems. An approximation method is also given which computes a lower bound on the solution and occasionally produces the exact solution.

The remaining sections investigate several topics related to the main theme. Section 7 examines various ways by which algorithms may be compared. The first is in terms of a forward analysis, where one algorithm is preferable to another if it can guarantee a smaller relative error bound for the solution. This method is equivalent to comparing, for each algorithm, the set of all computed solution relative error vectors with the unit cube. A second method, proposed by Miller [8], compares the sets of output relative error vectors for each algorithm against one another. The results of this comparison may indicate that in some sense each algorithm is preferable to the other. This dilemma is overcome by a third method which, like the first method, uses a common standard against which the algorithms are compared. Here, the standard is the problem which the algorithms solve and is represented by the set of output relative error vectors resulting from local relative errors in only input and output nodes. Several examples are given.

A composite algorithm is one which is made up of smaller sub-algorithms where the outputs of one sub-algorithm serve as the inputs to

the next. Section 8 shows how a composite algorithm may be analyzed if analyses have been performed on its parts. The relationship between the matrices of coefficients of the systems which relate local and total relative error in the composite algorithm and in the individual sub-algorithms forms the basis upon which the analysis of the composite algorithm may be done. Derivations are presented which provide bounds for forward, backward, and B-analysis of the composite algorithm in terms of corresponding analyses done on its sub-algorithms.

Finally, section 9 scratches the surface of the problem of algorithm improvement. The goal taken is to reduce the forward analysis error bounds, since by doing so bounds for other analyses should also be reduced. Algorithm improvement consists of two parts: locating where changes to the algorithm should be made, and deciding what changes should be incorporated. It is shown that operations that have large weights on the arcs leading from their nodes are prime candidates for modification. A change in the algorithm involving these nodes may take the form of a mathematical identity for which the best alternative may be easily computed. Several basic identities are investigated. This technique is limited in its effectiveness by the fact that the computational graph does not contain all the information about the algorithm or the problem it solves.

2. Preliminaries

This section presents the basic definitions and concepts provided by Bauer [4], and clarified and extended by Larson, Sameh [6].

Definition 2.1: problem - a set X of n input variables, ξ_i, $1 \leq i \leq n$, and a set Y of m output variables, η_i, $1 \leq i \leq m$, with $\eta_i = f_i(\xi_1, \xi_2, \ldots, \xi_n)$.

Definition 2.2: algorithm - any finite sequence of elementary operations of the form $\xi_i = \xi_j * \xi_k$, where * is one of $(+, -, \times, \div)$, which mathematically realize the elements of Y.

Definition 2.3: computational graph - a directed graph, essentially a parse tree of the algorithm, having the ξ_i as nodes and directed arcs from ξ_j and ξ_k to ξ_i.

The computational graph contains a total of r nodes consisting of n input nodes, ξ_i, $1 \leq i \leq n$, and r-n computational nodes, ξ_{n+i}, $1 \leq i \leq r-n$, of which m nodes are designated as output nodes, i.e., $\eta_i = \xi_{\omega_i}$, $1 \leq i \leq m$. The variable ω_i is used to index the outputs. The nodes are numbered in such a way so that if $\xi_i = \xi_j * \xi_k$, then $j < i$ and $k < i$. The value of each node is assumed to be non-zero.

The following model will be used from time to time to illustrate the concepts being presented.

$$\text{problem: } X = \{\xi_1, \xi_2, \xi_3, \xi_4\} \; ; \; n = 4,$$

$$Y = \{\eta_1, \eta_2\} \; ; \; m = 2,$$

$$\eta_1 = (\xi_1 + \xi_2) \div (\xi_2 - \xi_3),$$

$$\eta_2 = (\xi_2 - \xi_3) \times (\xi_3 + \xi_4).$$

algorithm: $\xi_5 = \xi_1 + \xi_2$

$\xi_6 = \xi_2 - \xi_3$

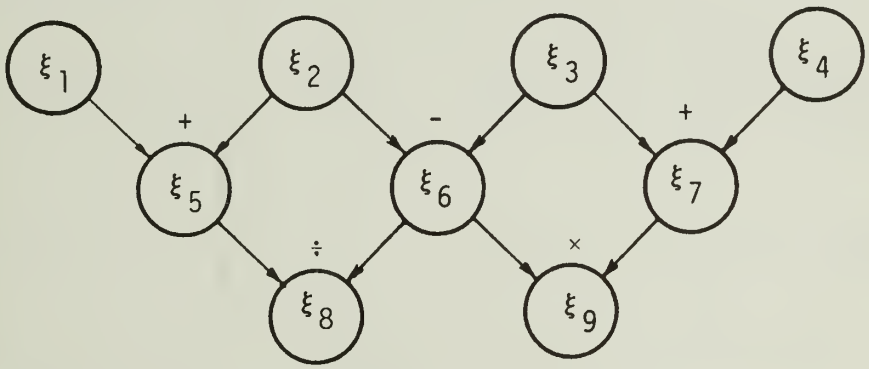
$\xi_7 = \xi_3 + \xi_4$

$\xi_8 = \xi_5 \div \xi_6$

$\xi_9 = \xi_6 \times \xi_7,$

where $\eta_1 = \xi_8$, $\eta_2 = \xi_9$, $\omega_1 = 8$, $\omega_2 = 9$, $r-n = 5$, $r = 9$.

computational graph:



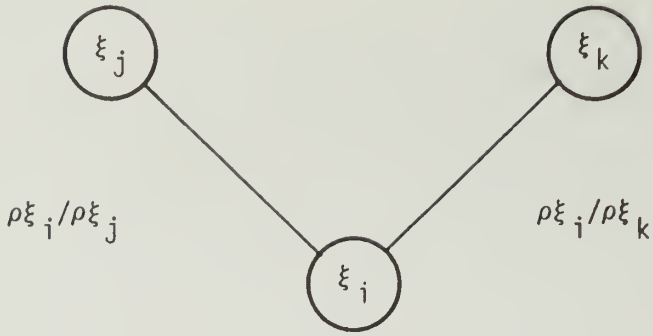
Subsequent graphs will employ implicit downward arrows.

Definition 2.4: partial relative derivative of ξ_i with respect to ξ_ℓ - a quantity, $\rho\xi_i/\rho\xi_\ell$, representing the first order relative effect on the value of ξ_i assuming a relative change in the value of ξ_ℓ, given by

$$\rho\xi_i/\rho\xi_\ell = (\xi_\ell/\xi_i)(\partial\xi_i/\partial\xi_\ell) \qquad (2.1)$$

Sterbenz [17] calls these quantities magnification factors.

The arcs of the computational graph are weighted. The weight of the arc from ξ_j to ξ_i is $\rho\xi_i/\rho\xi_j$. Table 1 illustrates the calculation of arc weights for the elementary operations.



$$\xi_i = \xi_j * \xi_k$$

	* = +	* = -	* = ×	* = ÷
$\rho\xi_i/\rho\xi_j$ =	ξ_j/ξ_i	ξ_j/ξ_i	1	1
$\rho\xi_i/\rho\xi_k$ =	ξ_k/ξ_i	$-\xi_k/\xi_i$	1	-1

Table 1. Arc Weights

The calculation of $\rho\xi_i/\rho\xi_\ell$ using (2.1) when $\ell \neq j$, $\ell \neq k$, or $\ell = j = k$ is complicated by the evaluation of $\partial\xi_i/\partial\xi_\ell$ by the chain rule. An alternative calculation makes use of the following definition and theorem.

Definition 2.5: path product - the product of partial relative derivatives belonging to each arc on a directed path between two nodes.

Theorem 2.1: $\rho\xi_i/\rho\xi_\ell$ is equal to the sum of the path products for all paths from ξ_ℓ and ξ_i.

Proof: The chain rule may be stated as

$$\partial\xi_i/\partial\xi_\ell = \sum_{p=1}^{s} \prod_{q=1}^{u_p} \partial\xi_{f(p,q)}/\partial\xi_{f(p,q+1)} \qquad (2.2)$$

where s is the number of distinct directed paths from ξ_ℓ to ξ_i; u_p is the number of arcs in path p; u_p+1 is the number of nodes in path p; $f(p,q) = f_{i\ell}(p,q)$ is a function which describes the paths from ξ_ℓ to ξ_i in the following manner: for each path p: $f(p,1) = i$; $\xi_{f(p,q+1)}$ is an operand of $\xi_{f(p,q)}$; $f(p,u_p+1) = \ell$. Consider two adjacent nodes, say $\xi_{f(p,q+1)}$ and $\xi_{f(p,q)}$, on one of these paths. From (2.1), the weight of the arc between these nodes is

$$\rho\xi_{f(p,q)}/\rho\xi_{f(p,q+1)} = (\xi_{f(p,q+1)}/\xi_{f(p,q)})(\partial\xi_{f(p,q)}/\partial\xi_{f(p,q+1)}).$$

Thus,

$$\partial\xi_{f(p,q)}/\partial\xi_{f(p,q+1)} = (\xi_{f(p,q)}/\xi_{f(p,q+1)})(\rho\xi_{f(p,q)}/\rho\xi_{f(p,q+1)}). \quad (2.3)$$

Substituting (2.3) in (2.2) gives the following expression for the partial relative derivative of ξ_i with respect to ξ_ℓ:

$$\rho\xi_i/\rho\xi_\ell = (\xi_\ell/\xi_i)\left(\sum_{p=1}^{s}\prod_{q=1}^{u_p}(\xi_{f(p,q)}/\xi_{f(p,q+1)})(\rho\xi_{f(p,q)}/\rho\xi_{f(p,q+1)})\right).$$

Simplification by cancellation results in

$$\rho\xi_i/\rho\xi_\ell = \sum_{p=1}^{s}\prod_{q=1}^{u_p}\rho\xi_{f(p,q)}/\rho\xi_{f(p,q+1)} \quad (2.4)$$

Thus, $\rho\xi_i/\rho\xi_\ell$ is the sum of all path products. ■

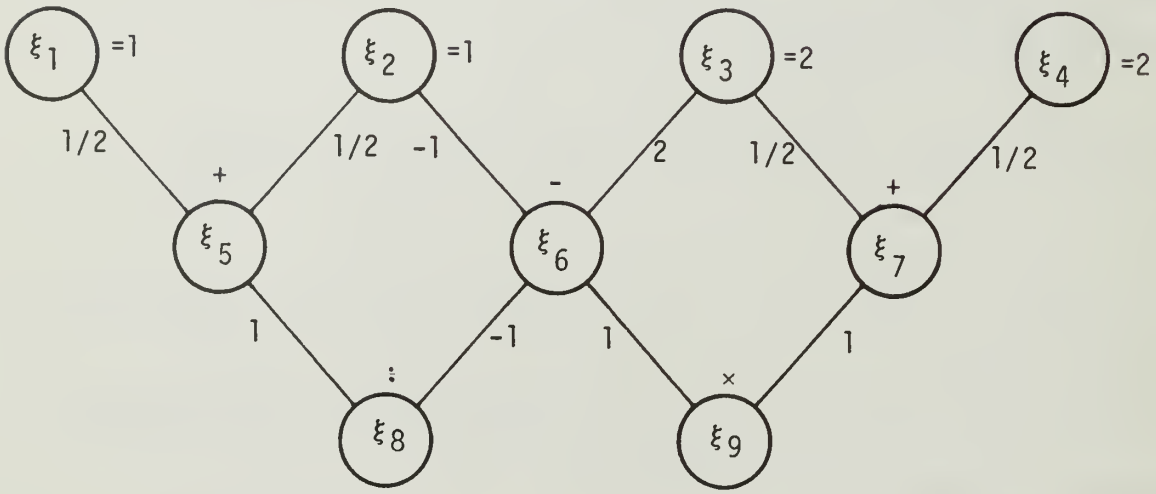
Trivially, $\rho\xi_i/\rho\xi_i = 1$, and if there is no directed path from ξ_ℓ to ξ_i, then $\rho\xi_i/\rho\xi_\ell = 0$.

Formula (2.4) may also be expressed recursively. If $\xi_i = \xi_j * \xi_k$, then

$$\rho\xi_i/\rho\xi_\ell = (\rho\xi_i/\rho\xi_j)(\rho\xi_j/\rho\xi_\ell) + (\rho\xi_i/\rho\xi_k)(\rho\xi_k/\rho\xi_\ell) \qquad (2.5)$$

This follows from the fact that any path from ξ_ℓ to ξ_i must pass through ξ_j or ξ_k.

For the model problem with data $\xi_1 = \xi_2 = 1$, $\xi_3 = \xi_4 = 2$, the arc weights are shown below.



The evaluation of $\rho\xi_9/\rho\xi_3$ using (2.1) necessitates the calculation

$$\partial\xi_9/\partial\xi_3 = \xi_2 - 2\xi_3 - \xi_4$$

from which follows

$$\rho\xi_9/\rho\xi_3 = (\xi_3/\xi_9)(\partial\xi_9/\partial\xi_3) = 5/2.$$

Alternatively, formula (2.4) gives

$$\rho\xi_9/\rho\xi_3 = (\rho\xi_9/\rho\xi_6)(\rho\xi_6/\rho\xi_3) + (\rho\xi_9/\rho\xi_7)(\rho\xi_7/\rho\xi_3) = 5/2.$$

3. Roundoff Errors

Floating point computation is subject to error due to finite word length. Wilkinson [19] provides the bounds for representation and arithmetic.

$$\bar{\xi}_i = fl(\xi_i) = \begin{cases} \xi_i(1 + \epsilon_i) & , 1 \leq i \leq n \\ (\xi_j * \xi_k)(1 + \epsilon_i), & n < i \leq r \end{cases}$$

where $|\epsilon_i| \leq \epsilon = (\beta^{1-d})/2$ (rounding with base β and mantissa of d digits.)

Definition 3.1: local relative error in ξ_i - a quantity, $[\rho\xi_i]_L \cdot \epsilon = \epsilon_i$, representing the relative change in the value of ξ_i due to its representation or computation.

It follows immediately that $|[\rho\xi_i]_L| \leq 1$.

Definition 3.2: total relative error in ξ_i - a quantity, $[\rho\xi_i]_T \cdot \epsilon$, representing the relative change in the value of ξ_i due to the accumulated effects of local relative errors in nodes of which ξ_i is a function.

Trivially, for input nodes, $[\rho\xi_i]_T \cdot \epsilon = [\rho\xi_i]_L \cdot \epsilon$, $1 \leq i \leq n$.

For a given algorithm, let $\xi_i = f(\xi_1, \xi_2, \ldots, \xi_{i-1})$, $n < i \leq r$. Previous local relative errors, $\epsilon_\ell = [\rho\xi_\ell]_L \cdot \epsilon$, $1 \leq \ell \leq i-1$, give rise to $\bar{\xi}_i = f(\xi_1(1 + \epsilon_1), \xi_2(1 + \epsilon_2), \ldots, \xi_{i-1}(1 + \epsilon_{i-1}))$. Expanding $\bar{\xi}_i$ around ξ_i using Taylor's theorem, and dividing by ξ_i results in

$$(\bar{\xi}_i - \xi_i)/\xi_i = \sum_{\ell=1}^{i-1} (\xi_\ell/\xi_i)(\partial\xi_i/\partial\xi_\ell)[\rho\xi_\ell]_L \cdot \epsilon + O(\epsilon^2) \qquad (3.1)$$

Adding $[\rho\xi_i]_L \cdot \epsilon$ to both sides of (3.1) and substituting (2.1), obtain

$$[\rho\xi_i]_T \cdot \epsilon = \sum_{\ell=1}^{i} (\rho\xi_i/\rho\xi_\ell)[\rho\xi_\ell]_L \cdot \epsilon + O(\epsilon^2)$$

Dividing by ϵ, and taking the limit as ϵ approaches 0 gives

$$[\rho\xi_i]_T = \sum_{\ell=1}^{i} (\rho\xi_i/\rho\xi_\ell)[\rho\xi_\ell]_L \tag{3.2}$$

By working with (3.2) the discussion will be independent of ϵ. Thus, the concern will only be with the coefficients of any small $\epsilon > 0$, and explicit mention of ϵ is henceforth dropped.

The formulation of total relative error, (3.2), gives rise to an undetermined homogeneous system of equations, $Mz = 0$, or

$$[-I_{r-n}, A, B] \begin{bmatrix} t \\ d \\ g \end{bmatrix} = 0 \tag{3.3}$$

where I_{r-n} is the identity matrix of order r-n; A is an $(r-n)\times n$ matrix with elements $a_{ij} = \rho\xi_{n+i}/\rho\xi_j$; B is an $(r-n)\times(r-n)$ unit lower triangular matrix with elements $\beta_{ij} = \rho\xi_{n+i}/\rho\xi_{n+j}$ below the diagonal; t is an (r-n)-vector with elements $\tau_i = [\rho\xi_{n+i}]_T$; d is an n-vector with elements $\delta_i = [\rho\xi_i]_L$; and finally, g is an (r-n)-vector with elements $\gamma_i = [\rho\xi_{n+i}]_L$.

System (3.3) may be used to perform an error analysis, e.g., forward, backward, etc., as in a following chapter. However, these analyses seek a relationship of errors between inputs and outputs, and between computational nodes and outputs. Hence, only those equations of (3.3) corresponding to output nodes need be considered, i.e., for $1 \leq i \leq m$,

$$[\rho\eta_i]_T = \sum_{\ell=1}^{\omega_i} (\rho\eta_i/\rho\xi_\ell)[\rho\xi_\ell]_L \tag{3.4}$$

where $\eta_i = \xi_{\omega_i}$. This produces the condensed system, $\overline{M}\,\overline{z} = 0$, or

$$[-I_m, \bar{A}, \bar{B},] \begin{bmatrix} \bar{t} \\ d \\ g \end{bmatrix} = 0 \qquad (3.5)$$

where I_m is the identity matrix of order m; $\bar{A}$ is an m×n matrix with elements $\bar{\alpha}_{ij} = \rho\eta_i/\rho\xi_j$; $\bar{B}$ is an m×(r-n) matrix with elements $\bar{\beta}_{ij} = \rho\eta_i/\rho\xi_{n+j}$; $\bar{t}$ is an m-vector with elements $\bar{\tau}_i = [\rho\eta_i]_T$; and d, and g are as previously defined.

The elements of M may be calculated from the formula for the partial relative derivative in (2.5). An obvious method forms the entries row by row, the partial relative derivatives on the right-hand side of (2.5) having already been computed. The matrix $\bar{M}$ can then be formed by selecting rows of A and B in M corresponding to output nodes, and appending $-I_m$. Since m is usually much less than r-n, i.e., few of the computational nodes are actually output nodes, much of the work done in forming M is concerned with rows which will eventually be discarded. An equivalent system which will allow the direct calculation of $\bar{M}$ will now be derived.

Formula (3.2) may be expressed as

$$[\rho\xi_i]_T = [\rho\xi_i]_L + \sum_{\ell=1}^{i-1} (\rho\xi_i/\rho\xi_\ell)[\rho\xi_\ell]_L$$

Substituting (2.5) gives

$$[\rho\xi_i]_T = [\rho\xi_i]_L + (\rho\xi_i/\rho\xi_j) \sum_{\ell=1}^{i-1} (\rho\xi_j/\rho\xi_\ell)[\rho\xi_\ell]_L$$

$$+ (\rho\xi_i/\rho\xi_k) \sum_{\ell=1}^{i-1} (\rho\xi_k/\rho\xi_\ell)[\rho\xi_\ell]_L$$

where $\xi_i = \xi_j * \xi_k$. From the node numbering scheme, $(\rho\xi_j/\rho\xi_\ell) = 0$, if $\ell > j$, and $(\rho\xi_k/\rho\xi_\ell) = 0$, if $\ell > k$; i.e., a node cannot be affected by a computation which is performed after its execution. Hence, changing the upper limits of both sums and substituting (3.2) results in

$$[\rho\xi_i]_T = [\rho\xi_i]_L + (\rho\xi_i/\rho\xi_j)[\rho\xi_j]_T + (\rho\xi_i/\rho\xi_k)[\rho\xi_k]_T \qquad (3.6)$$

In (3.6), the effects of previous local relative errors have been accumulated in the operands of ξ_i. Formula (3.6) is a local relationship of errors, since it concerns only a node and its operands. In contrast, one may call (3.2) a global relationship, since all previous nodes are inspected.

Thus, formula (3.6) produces the system $\tilde{M}z = 0$, or

$$[\tilde{B},\tilde{A},I_{r-n}] \begin{bmatrix} t \\ d \\ g \end{bmatrix} = 0 \qquad (3.7)$$

where $\tilde{B}$ is an $(r-n)\times(r-n)$ lower triangular matrix with elements $\tilde{\beta}_{ij} = \rho\xi_{n+i}/\rho\xi_{n+j}$, if ξ_{n+j} is an operand of ξ_{n+i}, and zero otherwise, and $\tilde{\beta}_{ii} = -1$; $\tilde{A}$ is an $(r-n)\times n$ matrix with elements $\tilde{\alpha}_{ij} = \rho\xi_{n+i}/\rho\xi_j$, if ξ_j is an operand of ξ_{n+i}, and zero otherwise; and I_{r-n}, and z are as previously defined.

The analyses could use either $Mz = 0$ or $\tilde{M}z = 0$. The reduction of $Mz = 0$ to $\bar{M}\,\bar{z} = 0$ is done to economize time and storage. No such reduction is possible with $\tilde{M}z = 0$, since this system specifies relationships at a local level, and to remove rows from it would result in the loss of the global relationships which the analyses seek.

Both systems specify a relationship between any and all errors, and their effects. The relationship specified in each system is the same,

since by comparing (3.3) and (3.7) and noting that $\tilde{B}$ is nonsingular, it must be that

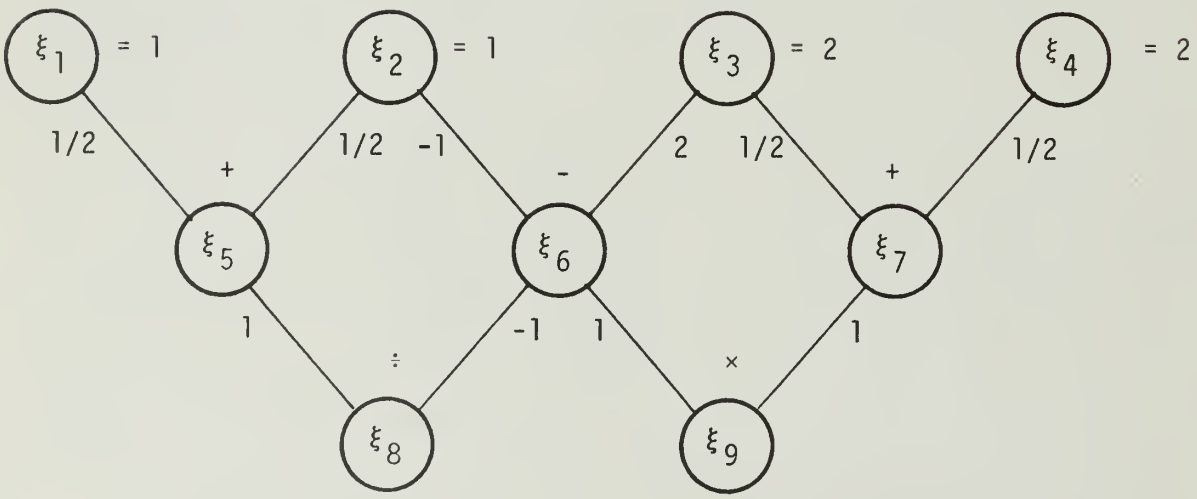
$$M = (-\tilde{B})^{-1} \tilde{M} \tag{3.8}$$

The systems $Mz = 0$, $\tilde{M}z = 0$, and $\overline{M}\,\overline{z} = 0$ for the model are given below. Row and column headings are provided as an aid in identifying matrix elements. The equation relating $[\rho\xi_9]_T$ to other errors is shown to illustrate the difference in complexity among the systems.

	-I					A				B				
	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9
ξ_5	-1	0	0	0	0	1/2	1/2	0	0	1	0	0	0	0
ξ_6	0	-1	0	0	0	0	-1	2	0	0	1	0	0	0
ξ_7	0	0	-1	0	0	0	0	1/2	1/2	0	0	1	0	0
ξ_8	0	0	0	-1	0	1/2	3/2	-2	0	1	-1	0	1	0
ξ_9	0	0	0	0	-1	0	-1	5/2	1/2	0	1	1	0	1

$$\begin{bmatrix} [\rho\xi_5]_T \\ [\rho\xi_6]_T \\ [\rho\xi_7]_T \\ [\rho\xi_8]_T \\ [\rho\xi_9]_T \\ \hline [\rho\xi_1]_L \\ [\rho\xi_2]_L \\ [\rho\xi_3]_L \\ [\rho\xi_4]_L \\ \hline [\rho\xi_5]_L \\ [\rho\xi_6]_L \\ [\rho\xi_7]_L \\ [\rho\xi_8]_L \\ [\rho\xi_9]_L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$Mz = 0$$



$$[\rho\xi_9]_T = -[\rho\xi_2]_L + 5/2[\rho\xi_3]_L + 1/2[\rho\xi_4]_L + [\rho\xi_6]_L + [\rho\xi_7]_L + [\rho\xi_9]_L$$

	$\tilde{B}$					$\tilde{A}$				I				
	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9
ξ_5	-1	0	0	0	0	1/2	1/2	0	0	1	0	0	0	0
ξ_6	0	-1	0	0	0	0	-1	2	0	0	1	0	0	0
ξ_7	0	0	-1	0	0	0	0	1/2	1/2	0	0	1	0	0
ξ_8	1	-1	0	-1	0	0	0	0	0	0	0	0	1	0
ξ_9	0	1	1	0	-1	0	0	0	0	0	0	0	0	1

$$\tilde{M}z = 0$$

$$\begin{bmatrix} [\rho\xi_5]_T \\ [\rho\xi_6]_T \\ [\rho\xi_7]_T \\ [\rho\xi_8]_T \\ [\rho\xi_9]_T \\ \hline [\rho\xi_1]_L \\ [\rho\xi_2]_L \\ [\rho\xi_3]_L \\ [\rho\xi_4]_L \\ \hline [\rho\xi_5]_L \\ [\rho\xi_6]_L \\ [\rho\xi_7]_L \\ [\rho\xi_8]_L \\ [\rho\xi_9]_L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$[\rho\xi_9]_T = [\rho\xi_9]_L + [\rho\xi_6]_T + [\rho\xi_7]_T$$

	-I		$\overline{A}$				$\overline{B}$				
	ξ_8	ξ_9	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9
η_1	-1	0	1/2	3/2	-2	0	1	-1	0	1	0
η_2	0	-1	0	-1	5/2	1/2	0	1	1	0	1

$$\overline{M}\,\overline{z} = 0$$

$$\begin{bmatrix} [\rho\eta_1]_T \\ [\rho\eta_2]_T \\ \hline [\rho\xi_1]_L \\ [\rho\xi_2]_L \\ [\rho\xi_3]_L \\ [\rho\xi_4]_L \\ \hline [\rho\xi_5]_L \\ [\rho\xi_6]_L \\ [\rho\xi_7]_L \\ [\rho\xi_8]_L \\ [\rho\xi_9]_L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$[\rho\xi_9]_T$ is as given for Mz = 0.

4. Efficient Calculation of $\bar{M}$

From (3.8), the relationship between M and $\tilde{M}$, one can devise an efficient scheme for forming $\bar{A}$ and $\bar{B}$ directly from $\tilde{A}$ and $\tilde{B}$ without forming A and B. Let $\bar{C} = [\bar{A},\bar{B}]$, $C = [A,B]$, $\bar{c}_i^t$ be the i^{th} row of $\bar{C}$ and the j^{th} row of C, where $n+j = \omega_i$. Since $C = (-\tilde{B})^{-1}[\tilde{A},I_{r-n}]$, it follows that

$$\bar{c}_i^t = e_j^t C = e_j^t(-\tilde{B})^{-1}[\tilde{A},I_{r-n}],$$

where e_j is the j^{th} column of the identity matrix of order r-n. Post-multiplying this equation by the nonsingular matrix,

$$\begin{bmatrix} I_n & 0 \\ -\tilde{A} & -\tilde{B} \end{bmatrix}$$

and transposing gives

$$\begin{bmatrix} I_n & -\tilde{A}^t \\ 0 & -\tilde{B}^t \end{bmatrix} \bar{c}_i = e'_{n+j},$$

where e'_{n+j} is the $n+j^{th}$ column of the identity matrix of order r. This unit upper triangular system may be solved in an efficient manner by drawing on the sparseness of $\tilde{A}$ and $\tilde{B}$. Let R be the coefficient matrix, $s = \bar{c}_i$, and $f = e'_{n+j}$. Then initially,

$$\begin{bmatrix} R_{r-1} & u_r \\ & 1 \end{bmatrix} \begin{bmatrix} s_{r-1} \\ \sigma_r \end{bmatrix} = \begin{bmatrix} f_{r-1} \\ \phi_r \end{bmatrix}$$

The value of σ_r is immediately known, and only the deflated system

$$R_{r-1}s_{r-1} = f_{r-1} - \sigma_r u_r$$

need be considered in order to find the remaining σ's. This system is of the same form as the original, and the same technique to find σ_{r-1} may be employed, etc. Each update of the right-hand side is particularly simple. At step k, $r \geq k \geq n+1$, the vector u_k^t is a subvector of the $(k-n)^{th}$ row of $[-\tilde{A},-\tilde{B}]$, and therefore is zero except for two elements. Let $u_k^t = (\mu_{k1}, \mu_{k2}, \ldots, \mu_{k,k-1})$. If $\xi_k = \xi_p * \xi_q$, then the non-zero elements of u_k^t are $\mu_{k,p} = -(\rho\xi_k/\rho\xi_p)$, and $\mu_{k,q} = -(\rho\xi_k/\rho\xi_q)$. Thus, the following algorithm computes the i^{th} row of $\bar{C}$.

1. $f^{(n+j)} = e'_{n+j}$, where $n+j = \omega_i$
2. For $k = n+j, n+j-1, \ldots, n+1$

$$f^{(k-1)} = f^{(k)},$$

$$\text{except } 1)\ \phi_p^{(k-1)} = \phi_p^{(k)} + \phi_k^{(k)}(\rho\xi_k/\rho\xi_p), \qquad (4.1)$$

$$2)\ \phi_q^{(k-1)} = \phi_q^{(k)} + \phi_k^{(k)}(\rho\xi_k/\rho\xi_q),$$

where $\xi_k = \xi_p * \xi_q$, and $f^{(k)} = (\phi_i^{(k)}), 1 \leq i \leq r$

3. $\bar{c}_i = f^{(n)}$

The algorithm starts with $f^{(n+j)}$ rather than $f^{(r)}$ since $\phi_i^{(r)} = 0$ for $i > n+j$, and hence $f^{(n+j)} = f^{(r)}$. The algorithm ends with $f^{(n)}$ rather than $f^{(1)}$ since after j steps $R_n = I_n$, and hence $f^{(n)} = f^{(1)}$.

An algorithm similar to (4.1) has been proposed by Linnainmaa [7] for the determination of Taylor coefficients for accumulated absolute rounding error.

The algorithm is related to formula (2.4) on path products. This relationship is based on the observation that any path from, say ξ_ℓ, to ξ_{n+j} must pass through some node ξ_s, where ξ_s has ξ_ℓ as one of its operands. Let S_ℓ be the set of all such indices s. Then (2.4) may be expressed as

$$\rho\xi_{n+j}/\rho\xi_\ell = \sum_{s\in S} (\rho\xi_{n+j}/\rho\xi_s)(\rho\xi_s/\rho\xi_\ell)$$

At each step k = s, $\phi_\ell^{(k)}$ is updated to include that path product corresponding to a path from ξ_ℓ to ξ_{n+j} through ξ_s, $\rho\xi_{n+j}/\rho\xi_s = \phi_k^{(k)}$ having been computed prior to step s.

The generation of $\bar{M}$ from $\tilde{M}$ takes considerably less time and storage than that for the generation of $\bar{M}$ from M. Computing the local partial relative derivatives in Table 1 requires r-n divisions, assuming each node type is equiprobable. These quantities are the $\tilde{\alpha}_{ij}$ and $\tilde{\beta}_{ij}$ of $\tilde{A}$ and $\tilde{B}$ in $\tilde{M}$. Algorithm (4.1) transforms $[\tilde{A},\tilde{B}]$ into $[\bar{A},\bar{B}]$ in no more than 4m(r-n) operations, depending on the distribution of the output nodes in the graph. In total, $\bar{M}$ is obtained from $\tilde{M}$ in (4m+1)(r-n) operations. The storage required is mr for $[\bar{A},\bar{B}]$, and 2(r-n) for a condensed form of $[\tilde{A},\tilde{B}]$, for a total of (m+2)r-2n.

Generating $\bar{M}$ from M again requires about r-n divisions for the local partial relative derivatives in Table 1. Following formula (2.5), each of the n(r-n) elements of A, and the $(r-n)^2/2$ elements of B uses 3 operations. Collectively, generating $\bar{M}$ from M requires (1 + (3r+n)/2)(r-n) operations. The storage required is r(r-n) for [A,B]. The elements of $[\tilde{A},\tilde{B}]$ are generated when needed and discarded, hence no additional storage is necessary.

By taking ratios, it follows that forming $\overline{M}$ from M requires approximately $3(r+n)/(8m)$ times the number of operations, and $(r-n)/(m+2)$ times the storage as that for generating $\overline{M}$ from $\tilde{M}$. For example, consider an algorithm in linear algebra with $n=k^2$ inputs, $m=k$ outputs, and $r-n=k^3$ computations, where k is the dimension of the problem. Then forming $\overline{M}$ from M requires $3(k^2+2k)/8$ the work and $k^3/(k+2)$ the storage of forming $\overline{M}$ from $\tilde{M}$.

These methods may be compared with existing ones, e.g., Miller [10], who has taken a different approach to error propagation. In his error analyses, the matrices A_p and A_r are generated, where A_p is an $(r-n)\times n$ matrix with elements $a_{ij}^{(p)} = \partial\xi_{n+i}/\partial\epsilon_j$; and A_r is an $(r-n)\times(r-n)$ lower triangular matrix with elements $a_{ij}^{(r)} = \partial\xi_{n+i}/\partial\epsilon_{n+j}$. These correspond to A and B of M. The correspondence is $\partial\xi_{n+i}/\partial\epsilon_\ell = \xi_{n+i}(\rho\xi_{n+i}/\rho\xi_\ell)$. By working with A_p and A_r, the assumption of non-zero node values may be dropped. Matrix elements are generated by the chain rule. If $\xi_{n+i} = \xi_j * \xi_k$, then the value of $\partial\xi_{n+i}/\partial\epsilon_\ell$ is given in Table 2.

	$\partial\xi_{n+i}/\partial\epsilon_\ell$
$* = +$	$\partial\xi_j/\partial\epsilon_\ell + \partial\xi_k/\partial\epsilon_\ell$
$* = -$	$\partial\xi_j/\partial\epsilon_\ell - \partial\xi_k/\partial\epsilon_\ell$
$* = \times$	$\xi_j(\partial\xi_k/\partial\epsilon_\ell) + \xi_k(\partial\xi_j/\partial\epsilon_\ell)$
$* = \div$	$(\xi_k(\partial\xi_j/\partial\epsilon_\ell) - \xi_k(\partial\xi_j/\partial\epsilon_\ell))/\xi_k^2$

Table 2. Evaluation of $\partial\xi_{n+i}/\partial\epsilon_\ell$

Note that $\partial\xi_{n+i}/\partial\epsilon_{n+i} = \xi_{n+i}$. Computation proceeds row by row, partial derivatives in the table having already been computed. Assuming each node type is equiprobable, one performs an average of 2.5 operations to evaluate

each of the (r+n)(r-n)/2 elements of A_p and A_r. Rows corresponding to output nodes are subsequently used to perform the analyses. The storage required is r(r-n) for $[A_p,A_r]$. In comparison to generating $\bar{M}$ from M, this technique takes 5/6 of the time and the same storage. With respect to $\bar{M}$ from $\tilde{M}$, it requires 5(r+n)/(16m) of the time and r(r-n)/(r(m+2)-2n) of the storage.

However, algorithm (4.1) may be modified to compute the elements of $[A_p,A_r]$. Since $\partial\xi_{n+i}/\partial\epsilon_\ell = \xi_\ell(\partial\xi_{n+i}/\partial\xi_\ell)$, the "path product" property of partial derivatives in (2.2) may be employed to compute the matrix elements to within a scalar factor. The modifications to algorithm (4.1) entail replacing $\rho\xi_k/\rho\xi_p$ by $\partial\xi_k/\partial\xi_p$, $\rho\xi_k/\rho\xi_q$ by $\partial\xi_k/\partial\xi_q$, and including a final scalar multiplication step. The modified algorithm would use Table 3 to calculate the arc weights. For $\xi_i = \xi_j * \xi_k$,

	* = +	* = -	* = ×	* = ÷
$\partial\xi_i/\partial\xi_j$	1	1	ξ_k	$1/\xi_k$
$\partial\xi_i/\partial\xi_k$	1	-1	ξ_j	$-\xi_i/\xi_k$

Table 3. Arc weights for Miller

While the computations required in Table 3 are less than that of Table 1, the final scalar multiplication step in the modified algorithm outweighs this and results in an overall increase of up to 25% over the time to execute algorithm (4.1). The storage required remains the same.

The generation of the equivalent of $\bar{M}$ in Miller [10] for several algorithms was timed on a 360/75. The results are summarized in Table 4. Many of the algorithms are described in Miller [12]. Givens/backsub reduces a matrix to upper triangular form using Givens plane rotations, and solves the resulting system by backsubstitution.

Algorithm	n	r-n	m	Miller [10]	Miller w/(4.1)	Speedup
Givens(QA=R)/backsub(Rx=Qf) (4×4)	20	172	4	0.98	0.085 sec.	11.5
Least Squares (4×3)	16	116	3	0.45	0.046	9.8
Givens(QA=R) (4×4)	16	120	10	0.49	0.092	5.3
Cholesky($A=LL^t$)($Ly=f$)($L^tx=y$) (4×4)	14	62	4	0.16	0.030	5.3
RINVERSE (4×4 UT)	10	30	10	0.042	0.010	4.2

Table 4. Timed Comparison

5. Error Analyses

5.1. Forward Analysis

A forward analysis finds a bound on the relative error in the solution $|[\rho\eta_i]_T|$, $1 \leq i \leq m$. Depending on the assumptions made, the relative error in η_i may be the result of inexact data, inexact arithmetic, or both. From the system $\bar{M}\,\bar{z} = 0$, (3.5),

$$\bar{t} = \bar{A}d + \bar{B}g \tag{5.1}$$

or, in terms of the matrix and vector elements, for $1 \leq i \leq m$,

$$[\rho\eta_i]_T = \sum_{j=1}^{n} (\rho\eta_i/\rho\xi_j)[\rho\xi_j]_L + \sum_{j=n+1}^{r} (\rho\eta_i/\rho\xi_j)[\rho\xi_j]_L \tag{5.2}$$

Considering for the moment any single output, η_i, the relative error in η_i can be maximized by choosing the local relative errors, $[\rho\xi_j]_L$, $1 \leq j \leq r$, in the following manner:

$$[\rho\xi_j]_L = \begin{cases} 1 & \text{if } \rho\eta_i/\rho\xi_j > 0 \\ c & |c| \leq 1, \text{ if } \rho\eta_i/\rho\xi_j = 0 \\ -1 & \text{if } \rho\eta_i/\rho\xi_j < 0 \end{cases} \tag{5.3}$$

This choice of the local relative errors guarantees that $[\rho\eta_i]_T$ takes on its maximum positive value. The maximum negative value of $[\rho\eta_i]_T$ has the same magnitude, and corresponds to changing the signs of $[\rho\xi_j]_L$ in (5.3). Without loss of generality, the positive value will be used.

The choice of local relative errors in (5.3) reduces (5.2) to

$$|[\rho\eta_i]_T| \leq \sum_{j=1}^{n} |\rho\eta_i/\rho\xi_j| + \sum_{j=n+1}^{r} |\rho\eta_i/\rho\xi_j| \qquad (5.4)$$

where a possibly different set of local relative errors is chosen for each η_i. Formula (5.4) shows that the relative error in η_i is bounded by $(\|\bar{a}_i\|_1 + \|\bar{b}_i\|_1)$ where $\bar{a}_i^t$ and $\bar{b}_i^t$ are the i^{th} row vectors of $\bar{A}$ and $\bar{B}$. The vector of relative error bounds, for all outputs, obtained from (5.4) is a pessimistic one in that, generally, each bound cannot be simultaneously attained.

Recall the matrices $\bar{A}$ and $\bar{B}$ for the model problem:

	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9
η_1	1/2	3/2	-2	0	1	-1	0	1	0
η_2	0	-1	5/2	1/2	0	1	1	0	1
	$\bar{A}$				$\bar{B}$				

Application of (5.4) gives a relative error bound of 7 for η_1 and corresponds to the choice of 1,1,-1,c,1,-1,c,1,c for the local relative errors. Similarly, the choice of c,-1,1,1,c,1,1,c,1 for $[\rho\xi_j]_L$, $1 \leq j \leq 9$, gives a relative error bound of 7 for η_2. Recall that these values are coefficients of ϵ.

Equation (5.1) shows the contributions of data error, d, and computational error, g, to the output error. By assuming exact computation, i.e., setting g = 0, then (5.1) reduces to $\bar{t} = \bar{A}d$, and (5.4) becomes

$$|[\rho\eta_i]_T| \leq \sum_{j=1}^{n} |\rho\eta_i/\rho\xi_j| \qquad (5.5)$$

Definition 5.1: condition vector of the problem - the vector of output relative error bounds as computed by (5.5). The elements of this vector

may be interpreted as the sensitivity of the outputs to input error.

For a problem with given data, the condition vector of the problem is independent of the algorithm. This fact follows from the assumption of exact computation. For the model problem with data $\xi_1 = \xi_2 = 1$, $\xi_3 = \xi_4 = 2$, the "condition vector of the problem" is always $(4,4)^t$.

By assuming exact data, i.e., d = 0, then $\overline{t} = \overline{B}g$, and

$$|[\rho\eta_i]_T| \leq \sum_{j=n+1}^{r} |\rho\eta_i/\rho\xi_j| \qquad (5.6)$$

Definition 5.2: condition vector of the algorithm - the vector of output relative error bounds as computed by (5.6). The elements of this vector may be interpreted as the sensitivity of the outputs to computational error.

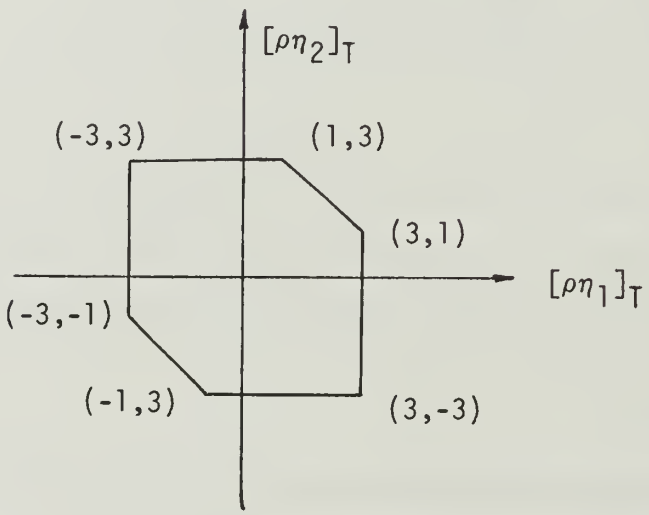
Naturally, the condition vector of the algorithm varies from algorithm to algorithm. For the model problem and the given data, the algorithm employed has a "condition vector of the algorithm" of $(3,3)^t$.

5.2. Backward Analysis

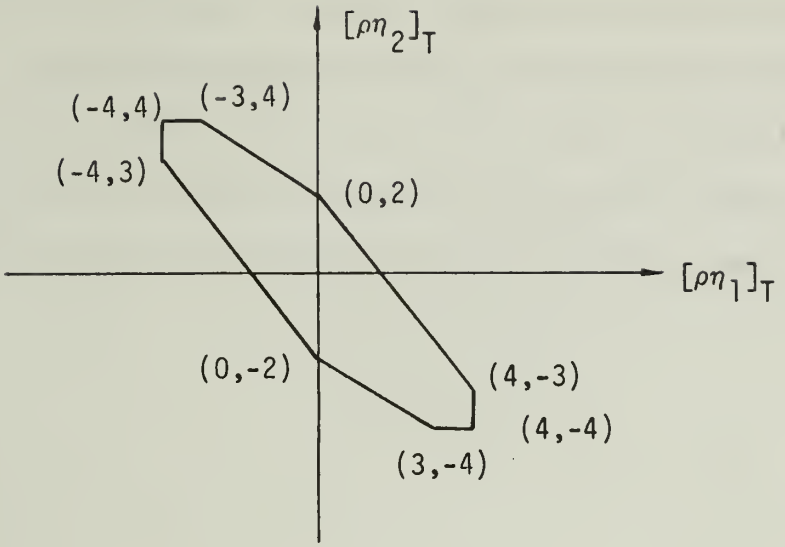
A backward analysis attempts to find a number, Θ, which measures the numerical stability of an algorithm. An algorithm is stable if it produces a solution which is the exact solution of the problem with data $\xi_i(1 + [\rho\xi_i]_L \cdot \epsilon)$, $1 \leq i \leq n$, where $|[\rho\xi_i]_L| \leq \Theta$.

The value of Θ is found by comparing the set of output relative error vectors corresponding to any computed solution with the set of output relative error vectors produced by some perturbation of the data. The analyses in this and the next section assume that the computed solution is obtained from exact data. Let $C_k = \{v: \|v\|_\infty \leq 1\}$ be the unit cube in R^k (the ∞-norm). Any vector, $g = (\gamma_i)$, $\gamma_i = [\rho\xi_{n+i}]_L$, $1 \leq i \leq r-n$, of

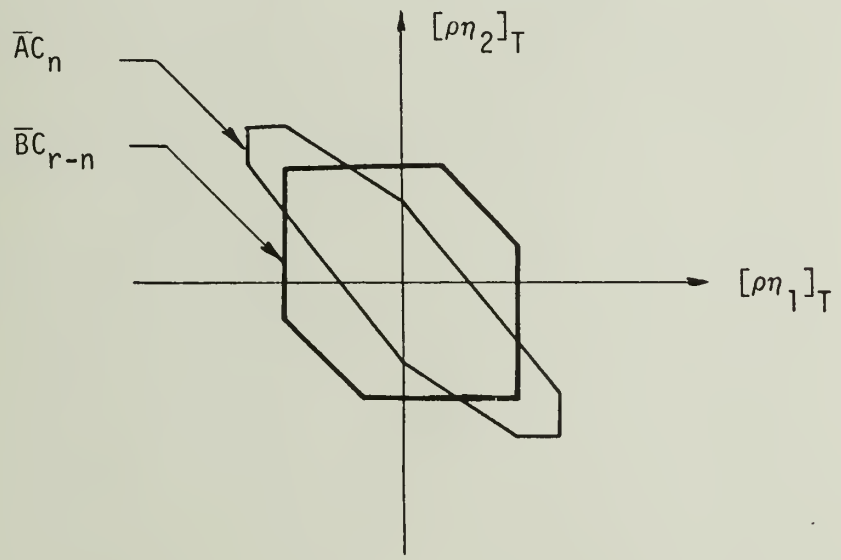
local relative errors in the computational nodes is an element of C_{r-n}, since $|\gamma_i| \leq 1$. From the forward analysis, a computed solution corresponding to a given g has an output relative error vector $\bar{B}g \in R^m$. It follows that any computed solution must have an output relative error vector which is an element of $\bar{B}C_{r-n} = \{\bar{B}g: g \in C_{r-n}\}$. The set, $\bar{B}C_{r-n}$ is a centrally symmetric convex polytope. This follows from the similar properties of C_{r-n}, and the fact that $\bar{B}$ is a linear transformation from R^{r-n} to R^m, Householder [5]. For the model problem with given data, the polytope $\bar{B}C_{r-n}$ is shown below.



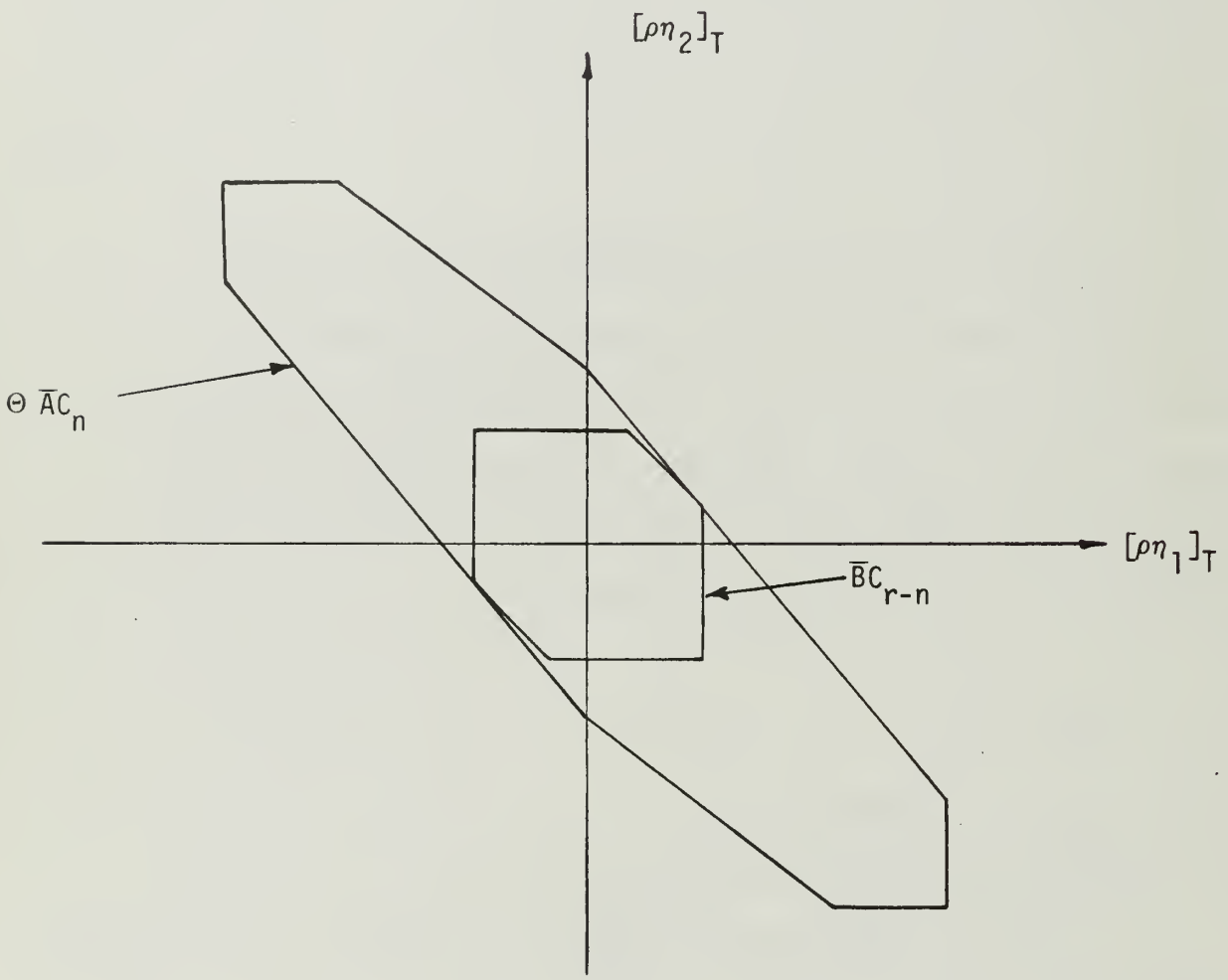
In a similar fashion the set of output relative error vectors produced by only perturbing the data may be found. Let $d = (\delta_i)$, $\delta_i = [\rho\xi_i]_L$, $1 \leq i \leq n$, be a vector of local relative errors in input nodes with $|\delta_i| \leq 1$. Then $d \in C_n$, and the corresponding output relative error vector for the solution is $\bar{A}d \in R^m$. The set of all such vectors forms $\bar{A}C_n = \{\bar{A}d: d \in C_n\}$, which is, again, a centrally symmetric convex polytope. For the model problem, the polytope $\bar{A}C_n$ is shown below.



To perform a backward analysis, the output relative error vector for any computed solution must be shown to be the same as one obtained by only perturbing the data. By superimposing $\overline{BC}_{r-n}$ and $\overline{AC}_n$ for the model problem, it is clear that there are output relative error vectors for computed solutions which are not output relative error vectors corresponding to input perturbations of only $1 \cdot \epsilon$.

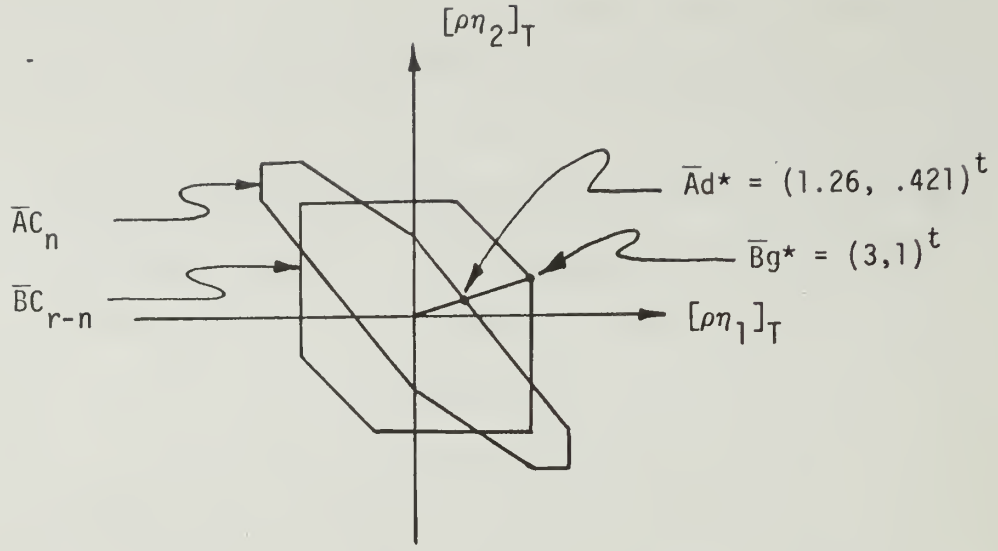


Thus, the value of $\Theta = 1$ is too small. The correct value of Θ is found by expanding (or contracting) $\bar{A}C_n$ so that it just covers $\bar{B}C_{r-n}$. This corresponds to allowing d to lie in ΘC_n, where $\Theta > 0$. $\bar{A}d$ is then an element of the polytope $\Theta\bar{A}C_n$. The value of Θ sought is $\text{glb}\{\Theta > 0: \bar{B}C_{r-n} \subseteq \Theta\bar{A}C_n\}$. For the model problem, $\Theta = 2.375$, as illustrated in the diagram below.



Thus, any computed solution of the model problem at the given data can be obtained by computing exactly with data $\xi_i(1 + [\rho\xi_i]_L \cdot \epsilon)$, $1 \leq i \leq 4$, where $|[\rho\xi_i]_L| \leq 2.375$.

The amount by which $\overline{AC}_n$ must be expanded to cover $\overline{BC}_{r-n}$ is governed by only a few of the elements of $\overline{BC}_{r-n}$. These elements correspond to the vertices of $\overline{BC}_{r-n}$. From the convex properties of $\overline{AC}_n$ and $\overline{BC}_{r-n}$, if $\overline{AC}_n$ covers the vertices of $\overline{BC}_{r-n}$, then it covers all of $\overline{BC}_{r-n}$. Hence, attention may be limited to covering only the vertices of $\overline{BC}_{r-n}$. As the previous diagram illustrates attention may, in fact, be restricted to covering only one vertex (and its symmetric mate). This vertex is the output relative error vector $\overline{B}g^*$ corresponding to a "worst" set g^* of local relative errors in the computational nodes. By covering this worst case, all others are covered. This vertex $\overline{B}g^*$ is characterized by being the "farthest" point on the perimeter, or boundary of $\overline{BC}_{r-n}$ from the boundary of $\overline{AC}_n$. Any point $\overline{B}g$ on the boundary of $\overline{BC}_{r-n}$ is covered, on expansion, by a corresponding point $\overline{A}d$ on the boundary of $\overline{AC}_{r-n}$. Both $\overline{B}g$ and the corresponding $\overline{A}d$ lie on the same radial line from the origin. Hence, $\overline{B}g = \Theta\overline{A}d$, where Θ measures the distance of $\overline{B}g$. For the model problem, the points $\overline{B}g^*$ and $\overline{A}d^*$ are illustrated below.



In general, there may be more than one symmetric pair of vertices on the boundary of $\bar{B}C_{r-n}$ whose relative distance from the boundary of $\bar{A}C_n$ is maximal. If this is the case, any arbitrary vertex of maximal distance may be used, since when any point on the boundary of $\bar{B}C_{r-n}$ of a given relative distance is covered, all points of that relative distance are also covered.

The chapter on computational details will discuss the finding of g*, and it will be assumed here that it has been computed. The following problem may be solved to obtain Θ.

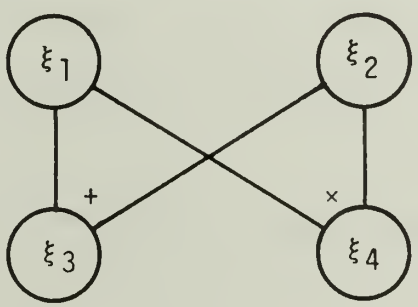
Backward analysis: find the smallest constant $\Theta > 0$, which satisfies the following conditions:

$$\begin{aligned} \bar{A}d &= \bar{B}g^* \\ |\delta_i| &\leq \Theta \qquad 1 \leq i \leq n \end{aligned} \tag{5.7}$$

where $d^t = (\delta_i)$, $\delta_i = [\rho\xi_i]_L$, $1 \leq i \leq n$, and g* is as described above. The minimal Θ is the ∞-norm of that vector $d \in \Theta C_n$ of input relative errors

whose image under $\bar{A}$ covers the "worst" output relative error vector $\bar{B}g^*$. $\Theta\bar{A}C_n$ therefore covers $\bar{B}C_{r-n}$.

Having set up the problem (5.7), it may be found to be unsolvable. From the construction of $\bar{B}$ it is easy to show that its columns span R^m. The polytope $\bar{B}C_{r-n}$ therefore extends in all m dimensions. If $m > n$, i.e., if there are more outputs than inputs, then the n columns of $\bar{A}$ can do no better than to span R^n. The polytope $\bar{A}C_n$ can therefore extend in no more than n dimensions. It is impossible to cover a polytope in R^m with another in R^n, where $m > n$. This deficiency is reflected in the inconsistency of the equations in (5.7), i.e., rank $(\bar{A})$ < rank $([\bar{A},\bar{B}g^*])$. Even if $m \leq n$, which is usually the case, there is no guarantee that m of the columns of $\bar{A}$ are linearly independent, and hence, that $\bar{A}C_n$ extends in m dimensions. Again, the situation could be the impossible task of covering a polytope in R^m with another in R^k, where $m > k$. The following example illustrates this case.

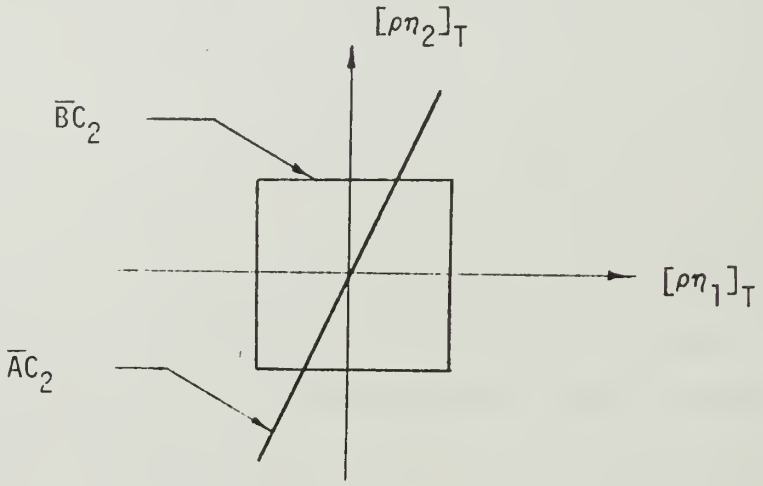


This is the computational graph for the algorithm: $\eta_1 = \xi_3 = \xi_1 + \xi_2$; $\eta_2 = \xi_4 = \xi_1 \times \xi_2$.

For $\xi_1 = \xi_2$,

$$\bar{A} = \begin{bmatrix} 1/2 & 1/2 \\ 1 & 1 \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The polytopes $\bar{A}C_2$ and $\bar{B}C_2$ are shown below.



The polytope $\bar{A}C_2$ consists of the line segment joining the points (-1, -2) and (1, 2). The polytope $\Theta\bar{A}C_2$ for any Θ will consist of the line segment joining the points $\Theta(-1, -2)$ and $\Theta(1, 2)$, and cannot cover the polytope $\bar{B}C_2$.

The inability to always perform a backward analysis warrants a relaxation of the criterion for stability to allow an analysis which may always be done. One such analysis is presented in the next section.

5.3. B-analysis

A forward and backward analysis can be combined into what Panzer [14] calls a B-(Beitseitige) analysis. The B-analysis is along the lines of Stewart's [18] definition of stability, namely, that the

computed solution is "near" the exact solution of a perturbed problem. From the forward analysis, any computed solution, assuming exact data, must have an output relative error vector

$$t_B = \bar{B}g$$

which is an element of $\bar{B}C_{r-n}$. Similarly, the output relative error vector produced by perturbing only the data and computing exactly is

$$t_A = \bar{A}d$$

which is an element of $\bar{A}C_n$. The backward analysis requires finding the smallest number Θ such that for any choice of g in C_{r-n},

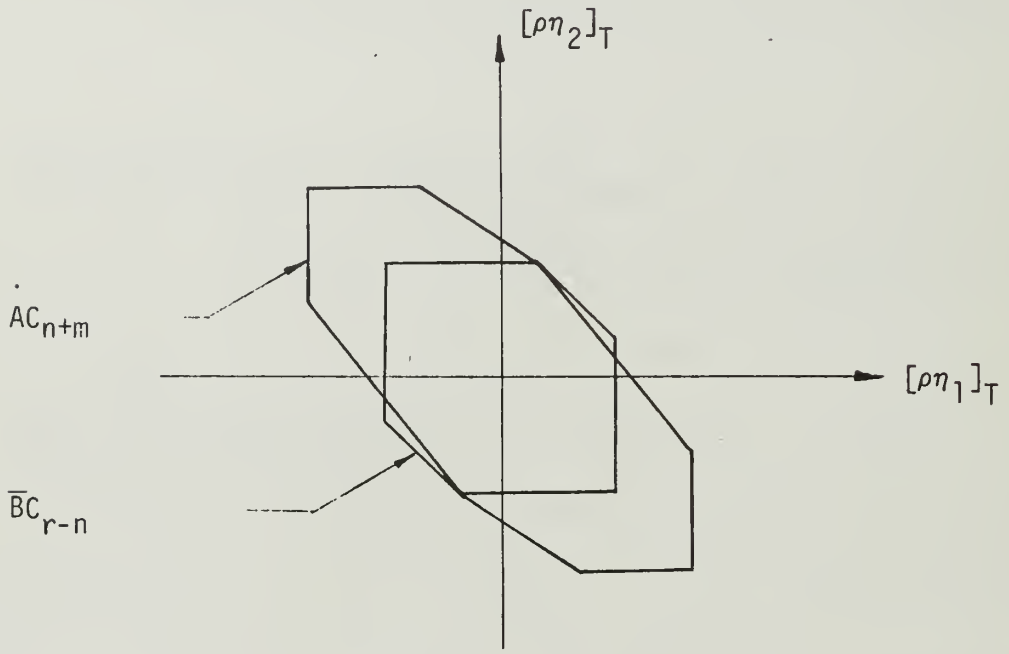
$$\Theta' t_A = t_B$$

where $\Theta' \leq \Theta$. However, the B-analysis only requires that $\Theta' t_A$ be "near" t_B. The "nearness" of $\Theta' t_A$ to t_B can be measured by the ∞-norm, Θ', of a vector $\Theta'\hat{t}$ such that

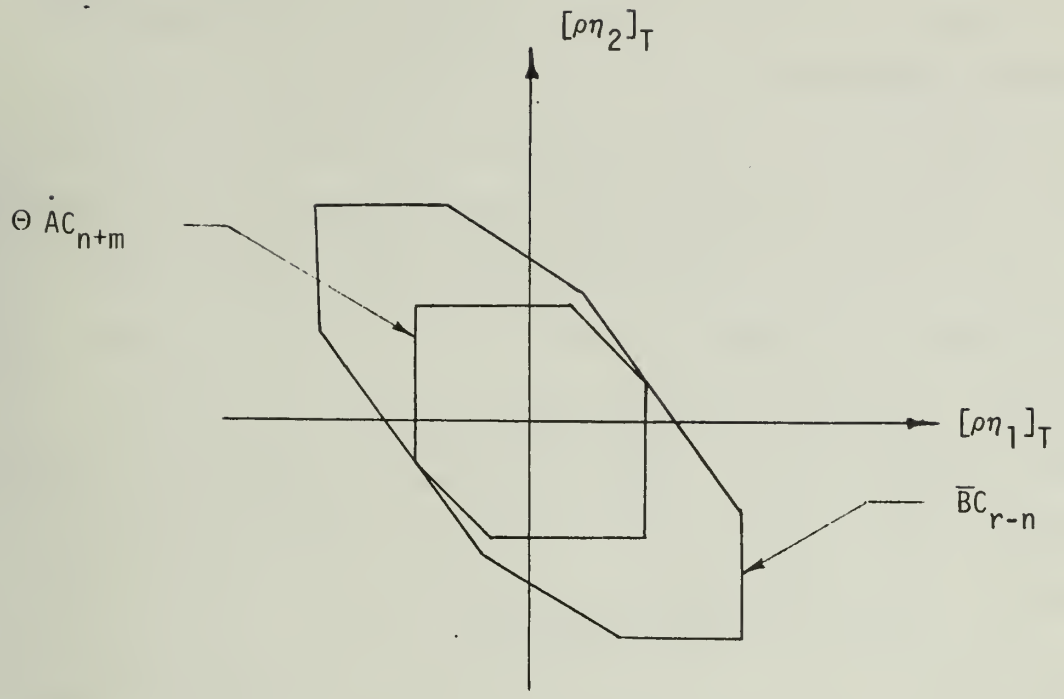
$$\Theta'(t_A + \hat{t}) = t_B \tag{5.8}$$

where $\hat{t} = (\hat{\tau}_i)$, $\hat{\tau}_i = [\rho\eta_i]$, $|\hat{\tau}_i| \leq 1$, $1 \leq i \leq m$, is a vector of relative errors which artificially perturb the output values. A constant, Θ, is now sought such that for any g in C_{r-n}, (5.8) holds for some $\Theta' \leq \Theta$. By bounding both the input and output perturbations by Θ, the "nearness" of the perturbed inputs to the true inputs, and the "nearness" of the computed solution to the exact solution of the perturbed problem are measured in the same way.

The value of Θ is found by comparing sets of output relative error vectors. As before, the set of all vectors, t_B, is $\overline{BC}_{r-n}$. Let $\dot{A} = [\overline{A}, I_m]$, and $\dot{d}^t = (d^t, \hat{t}^t)$. Then $t_A + \hat{t} = \dot{A}\dot{d}$, where $\dot{d} \in C_{n+m}$. The set of all vectors, $t_A + \hat{t}$, is $\dot{A}C_{n+m}$, a centrally symmetric, convex polytope. For the model problem, $\dot{A}C_{n+m}$ and $\overline{BC}_{r-n}$ are shown below.



The B-analysis now follows a strategy similar to that of the backward analysis. It must be shown that the output relative error vector for any computed solution is the same as that obtained by perturbing the data and the outputs. The polytope $\dot{A}C_{n+m}$ is expanded so that it just covers $\overline{BC}_{r-n}$. Thus, $\dot{d}$ is allowed to lie in ΘC_{n+m}, where $\Theta > 0$. $\dot{A}\dot{d}$ is then an element of $\Theta\dot{A}C_{n+m}$. The value of Θ sought is $\text{glb}\{\Theta > 0: \overline{BC}_{r-n} \subseteq \Theta\dot{A}C_{n+m}\}$. The diagram below shows that $\Theta = 1.1176$ is the desired Θ for the model problem.



Thus, any computed solution of the model problem at the given data can be obtained by computing exactly with data $\xi_i(1 + [\rho\xi_i]_L \cdot \epsilon)$, $1 \leq i \leq 4$, and then perturbing the outputs η_i by $(1 + [\rho\eta_i] \cdot \epsilon)$, $1 \leq i \leq 2$, where $|[\rho\xi_i]_L| \leq 1.1176$, and $|[\rho\eta_i]| \leq 1.1176$.

As in the backward analysis, Θ is governed by a vertex $\overline{B}g^*$ corresponding to a "worst" set g^* of local relative errors in the computational nodes. The vertex $\overline{B}g^*$ is one of the points on the boundary of $\overline{BC}_{r-n}$ of maximal relative distance from $\dot{AC}_{n+m}$, where distance is measured in the same way as that in the backward analysis. For the model problem, g^* is the same for both the backward and B-analysis, though, in general, this need not be the case.

Assuming that g^* has been computed, the following problem may

solved to obtain Θ.

- B-analysis: find the smallest constant $\Theta > 0$, which satisfies the following conditions:

$$\begin{aligned} \dot{A}\dot{d} &= \bar{B}g^* \\ |\dot{\delta}_i| &\leq \Theta \quad 1 \leq i \leq n+m \end{aligned} \tag{5.9}$$

where $\dot{d}^t = (\dot{\delta}_i)$, $1 \leq i \leq n+m$

$$\dot{\delta}_i = \begin{cases} \delta_i = [\rho\xi_i]_L \text{ for } 1 \leq i \leq n \\ \hat{\tau}_{i-n} = [\rho\eta_{i-n}] \text{ for } n+1 \leq i \leq n+m \end{cases}$$

$\dot{A} = [\bar{A}, I_m]$, and

g^* is as described previously.

The minimum Θ is the ∞-norm of that vector $\dot{d} \in \Theta C_{n+m}$ of input and output relative errors whose image under $\dot{A}$ covers the "worst" output relative error vector $\bar{B}g^*$. $\Theta\dot{A}C_{m+n}$ therefore covers $\bar{B}C_{r-n}$.

The system of equations in (5.9) is always underdetermined since $\dot{A}$ is an $m\times(m+n)$ matrix. Furthermore, a vector, $\dot{d}^t = (d^t, \hat{t}^t)$, which satisfies these equations can always be found since rank $(\dot{A})$ = rank $([\bar{A}, I_m])$ = m. In particular, the vector $(0, (\bar{B}g^*)^t)^t$ satisfies these equations, and corresponds to a forward analysis. This vector, however, does not usually give the smallest Θ.

6. Computational Details

6.1. The Form of g*

Before the problem for backward analysis (5.7) or the problem for B-analysis (5.9) can be solved, the vertex, $\bar{B}g^*$, of the computational polytope of maximal relative distance from the problem polytope must be found. The following discussion will concern the B-analysis to which obvious modifications can be made if a backward analysis is to be attempted.

Since $\bar{B}$ is fixed, the problem is that of finding g*. From the properties of $\bar{B}$ and C_{r-n}, g* is found to have a particular form as shown by the following argument. Any point, x, in a convex set may be represented by $x = \lambda a + (1-\lambda)b$ for some a and b in the set and $0 \leq \lambda \leq 1$. A vertex, x, has a unique representation of the form $x = 1 \cdot x$. Let x be an element of $\bar{B}C_{r-n}$, i.e., $x = \sum_{i=1}^{r-n} \gamma_i \bar{b}_i$, where $\bar{b}_i$ is the i^{th} column of $\bar{B}$ and $|\gamma_i| \leq 1$.

Theorem 6.1: If x is a vertex, then $|\gamma_i| = 1$, $1 \leq i \leq r-n$.

Proof: Assume there exists a j such that $|\gamma_j| < 1$. Without loss of generality, assume $0 \leq \gamma_j < 1$. An analogous proof holds for $-1 \leq \gamma_j < 0$. Then

$$x = \sum_{\substack{i=1 \\ i \neq j}}^{r-n} \gamma_i \bar{b}_i + \gamma_j \bar{b}_j$$

Let $u = \sum_{\substack{i=1 \\ i \neq j}}^{r-n} \gamma_i \bar{b}_i$. Then $x = u + \gamma_j \bar{b}_j$. If $\gamma_j = 0$, then $x = u$. However, $(u + \bar{b}_j)$ and $(u - \bar{b}_j)$ are elements of $\bar{B}C_{r-n}$. Hence,

$$x = \lambda(u - \bar{b}_j) + (1 - \lambda)(u + \bar{b}_j), \text{ where } \lambda = 1/2$$

Thus, x is a non-trivial convex combination of two points in $\bar{B}C_{r-n}$ which

contradicts the assumption that x was a vertex.

- If $0 < \gamma_j < 1$, then, again, $x = u + \gamma_j \bar{b}_j$. However, u and $(u + \bar{b}_j)$ are elements of $\bar{B}C_{r-n}$, and

$$x = \lambda u + (1 - \lambda)(u + \bar{b}_j), \text{ where } \lambda = 1 - \gamma_j$$

Again, the fact that x is a vertex is contradicted. Since the choice of γ_j was arbitrary, it must be that any vertex has the form

$$x = \sum_{i=1}^{r-n} \gamma_i \bar{b}_i, \text{ where } |\gamma_i| = 1 \qquad \blacksquare$$

Thus, g* must have components $\gamma_i^* = \pm 1$. Intuitively, one would expect a "worst" set of computational errors to contain a maximal error for each step of the algorithm. Thus, the problem of finding g* consists only of finding the signs of its components.

For the model problem, $g^* = (1,-1,1,1,1)^t$.

6.2. Simplification of Polytope Representation

For efficiency, it will be of advantage to reduce the number of columns in $\dot{A}$ and $\bar{B}$. The following argument shows how this may be done. Let v be a vector formed by the sum of j vectors, v_i, $1 \leq i \leq j$, which are in the same or opposite direction. Then

$$v = \sum_{i=1}^{j} a_i v_i, \text{ with } |a_i| \leq 1 \qquad (6.1)$$

where for $1 \leq i \leq j$,

$$v_i = (-1)^{\sigma_i} \beta_i v_1$$
$$\sigma_i = -\text{sign}\,(v_i^t v_1)$$
$$\beta_i = \|v_i\|_\infty / \|v_1\|_\infty > 0$$

Then v may be represented by

$$v = a\, w, \text{ with } |a| \leq 1 \tag{6.2}$$

where

$$a = \left(\sum_{i=1}^{j} a_i(-1)^{\sigma_i}\beta_i\right)/\left(\sum_{i=1}^{j}\beta_i\right) \tag{6.3}$$

$$w = \sum_{i=1}^{j}(-1)^{\sigma_i} v_i \tag{6.4}$$

Formulas (6.1) through (6.4) provide the basis for condensing $\bar{B}$ to a matrix $\hat{\bar{B}}$ having $k_1 \leq r-n$ columns, $\hat{\bar{b}}_j$. In a similar fashion, the matrix $\dot{A} = [\bar{A}, I_m]$ may be condensed to $\hat{\dot{A}} = [\hat{\bar{A}}, D]$ having $k_2+m \leq n+m$ columns, $\hat{\dot{a}}_j$. The matrix D replaces I_m allowing for the case where some of the columns of $\bar{A}$ are multiples of e_i, the columns of I_m.

Following (6.4), the matrix $\bar{B}$ is condensed to $\hat{\bar{B}}$ by post-multiplication by a $(r-n)\times k_1$ matrix P_B,

$$\hat{\bar{B}} = \bar{B}\, P_B \tag{6.5}$$

where P_B has elements

$$\pi^{B}_{ji} = \begin{cases} +1 \text{ if } \bar{b}_j \text{ is condensed into } \hat{\bar{b}}_i \\ -1 \text{ if } -\bar{b}_j \text{ is condensed into } \hat{\bar{b}}_i \\ 0 \text{ otherwise} \end{cases}$$

The matrix P_B has the following properties:

(1) each row has only one non-zero element,

(2) the columns are orthogonal,

(3) $P_B^t P_B = D_B$, a diagonal matrix whose i^{th} diagonal element

equals the number of columns of $\bar{B}$ which are condensed into $\hat{\bar{b}}_i$.

For the model problem (6.5) reveals

$$\begin{bmatrix} 2 & -1 & 0 \\ 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

These reductions have not changed the polytopes, and the following proof shows $\hat{\bar{B}}\, C_{k_1} = \bar{B}\, C_{r-n}$. A similar proof would show that $\dot{\hat{A}}\, C_{k_2+m} = \dot{A}\, C_{n+m}$.

Theorem 6.2: $\hat{\bar{B}}\, C_{k_1} = \bar{B}\, C_{r-n}$.

Proof: a) Let $x = \hat{\bar{B}}\, \hat{g}$, where $\hat{g} = (\hat{\gamma}_i)$, $1 \le i \le k_1$, with $|\hat{\gamma}_i| \le 1$. Then $x \in \hat{\bar{B}}\, C_{k_1}$. From (6.5),

$$x = \hat{\bar{B}}\, \hat{g} = \bar{B}\, P_B\, \hat{g}$$

Since P_B has only one non-zero element per row and this non-zero element has magnitude one, it follows that

$$\|P_B\, \hat{g}\|_\infty \le 1$$

Hence, there exists a $g \in C_{r-n}$ (namely $g = P_B\, \hat{g}$) such that $x = \bar{B}g$. Thus, $x \in \bar{B}\, C_{r-n}$, and $\hat{\bar{B}}\, C_{k_1} \subseteq \bar{B}\, C_{r-n}$.

b) Let $x = \bar{B}g$, where $g = (\gamma_i)$, $1 \le i \le r-n$, with $|\gamma_i| \le 1$. Then $x \in \bar{B}\, C_{r-n}$. Let the columns of $\bar{B}$ be partitioned according to the sets J_i, $1 \le i \le k_1$, where $j \in J_i$ if $\bar{b}_j$ or $-\bar{b}_j$ is condensed into $\hat{\bar{b}}_i$. Then

$$x = \sum_{j\in J_1} \gamma_j \bar{b}_j + \sum_{j\in J_2} \gamma_j \bar{b}_j + \ldots + \sum_{j\in J_{k_1}} \gamma_j \bar{b}_j$$

In each sum, $\sum_{j\in J_i} \gamma_j \bar{b}_j$, all the vectors, $\bar{b}_j$, are in the same or opposite direction, and $|\gamma_j| \leq 1$. Each sum has the form given by (6.1), and may be substituted by $\hat{\gamma}_i \hat{\bar{b}}_i$, where $\hat{\gamma}_i$ and $\hat{\bar{b}}_i$ are computed via formulas analogous to (6.3) and (6.4). The formula analogous to (6.4) is (6.5). The formula analogous to (6.3) is

$$\hat{g} = Q_B^t g$$

where Q_B^t is a $k_1 \times (r-n)$ matrix with elements

$$\kappa_{ij}^B = \begin{cases} \|\bar{b}_j\|_\infty / \|\hat{\bar{b}}_i\|_\infty & \text{if } \bar{b}_j \text{ is condensed into } \hat{\bar{b}}_i \\ -\|\bar{b}_j\|_\infty / \|\hat{\bar{b}}_i\|_\infty & \text{if } -\bar{b}_j \text{ is condensed into } \hat{\bar{b}}_i \\ 0 & \text{otherwise} \end{cases}$$

The matrix Q_B^t has the following properties:

(1) each column has only one non-zero element,

(2) the rows are orthogonal,

(3) each absolute row sum equals one. This follows from $\|\hat{\bar{b}}_i\|_\infty = \sum_{j\in J_i} \|\bar{b}_j\|_\infty$,

(4) $Q_B^t Q_B = D_B$, a diagonal matrix,

(5) Q_B^t has the same structure as P_B^t, i.e., corresponding elements of Q_B^t and P_B^t are either both zero, both positive, or both negative,

(6) $Q_B^t P_B = I_{k_1}$. This follows immediately from properties (3) and (5).

For the model problem, and $g^t = (1,-1,1,1,1)$, the corresponding $\hat{g}$ is given below.

$$\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Thus, performing the above substitutions, $x = \sum_{i=1}^{k_1} \hat{\gamma}_i \hat{\bar{b}}_i$, with $|\hat{\gamma}_i| \leq 1$. Hence, $x \in \hat{\bar{B}}\, C_{k_1}$, and $\bar{B}\, C_{r-n} \subseteq \hat{\bar{B}}\, C_{k_1}$. ∎

Similar arguments reveal that any point x in $\dot{A}\, C_{n+m}$ of the form $\dot{\dot{A}}d = \sum_{i=1}^{n+m} \dot{\delta}_i \dot{a}_i$ can be expressed as an element of $\hat{\dot{A}}\, C_{k_2+m}$ of the form $\hat{\dot{\dot{A}}}d = \sum_{i=1}^{k_2+m} \hat{\dot{\delta}}_i \hat{\dot{a}}_i$, and vise versa. Hence $\dot{A}\, C_{n+m} = \hat{\dot{A}}\, C_{k_2+m}$.

For the model problem

$$\hat{\dot{A}} = \begin{bmatrix} 3/2 & -2 & 3/2 & 0 \\ -1 & 5/2 & 0 & 3/2 \end{bmatrix}, \quad \hat{\bar{B}} = \begin{bmatrix} 2 & -1 & 0 \\ 1 & 0 & 2 \end{bmatrix},$$

where $k_2+m = 4$, and $k_1 = 3$. The following table shows the often dramatic reductions for a battery of problems and algorithms, see Appendix for description.

ALGORITHM	m	r-n	k_1	n+m	k_2+m
BACK4	3	12	3	13	6
SEMI4	3	12	3	13	6
PARA4	3	14	4	13	6
BACK8	7	56	7	43	15
SEMI8	7	56	7	43	15
PARA8	7	86	11	43	15
GS	12	81	26	24	19
MGS	12	81	28	24	19
LTBAND	7	86	10	37	15
RINVERSE	10	30	10	20	19
GJ	4	76	7	24	8
GE	4	62	6	24	8
CHOL	4	62	13	18	14
MATMUL	16	112	16	48	48
CG	4	122	32	12	9

Table 5. Simplification of Polytope Representation

As the table illustrates, the reduction of $\overline{B}$ to $\hat{\overline{B}}$ occasionally results in k_1=m in which case $\hat{\overline{B}}$ is, to within column permutation, a lower triangular matrix. The following theorem gives sufficient conditions for this to occur.

Theorem 6.3: If the computational graph is a binary tree, i.e., each node is an operand of no more than one node, then $\hat{\bar{B}}$ is lower triangular.

Proof: For each node η_j, $1 \leq j \leq m$, define

$$S_j = \{\xi_{n+i}: \rho\eta_j/\rho\xi_{n+i} \neq 0 \text{ and } \rho\eta_k/\rho\xi_{n+i} = 0 \text{ for all } k < j\}$$

A node belongs to S_j if η_j is the first output node encountered on the unique path from it to the root of the tree, η_m. Every computational node appears in one and only one set, and each set is non-empty, since η_j belongs to S_j. Let the columns of $\bar{B}$ be partitioned according to the sets S_j, i.e., column i is in partition block j if ξ_{n+i} is in S_j. Consider the columns corresponding to $S_1 = \{\xi_{n+i}: \rho\eta_1/\rho\xi_{n+i} \neq 0\}$. For any ξ_{n+i} in S_1, the corresponding column in $\bar{B}$ is

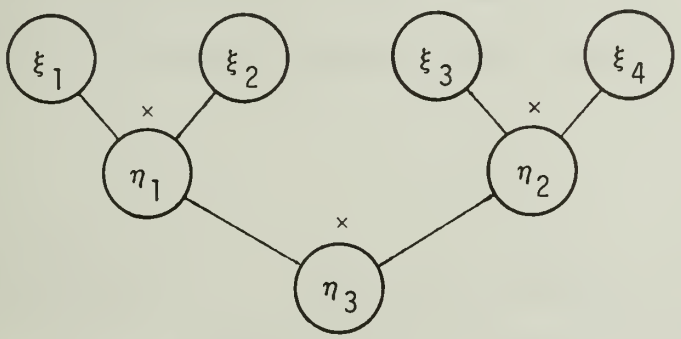
$$(\rho\eta_1/\rho\xi_{n+i}, \rho\eta_2/\rho\xi_{n+i}, \cdots, \rho\eta_m/\rho\xi_{n+i})^t = v_1 \cdot (\rho\eta_1/\rho\xi_{n+i})$$

where $v_1 = (\rho\eta_1/\rho\eta_1, \rho\eta_2/\rho\eta_1, \cdots, \rho\eta_m/\rho\eta_1)^t$. This follows from the fact that the graph is a tree and hence $\rho\eta_j/\rho\xi_{n+i} = (\rho\eta_j/\rho\eta_1)(\rho\eta_1/\rho\xi_{n+i})$, since there is at most one path from ξ_{n+i} to η_j. Thus, all the columns of $\bar{B}$ corresponding to S_1 are multiples of v_1, and the reduction of $\bar{B}$ may choose the first column of $\hat{\bar{B}}$ to be $v_1 \cdot \sum_{\xi_{n+i} \in S_1} |\rho\eta_1/\rho\xi_{n+i}|$. Following similar arguments, the column of $\bar{B}$ corresponding to ξ_{n+i} in S_j is $v_j \cdot (\rho\eta_j/\rho\xi_{n+i})$ where

$$v_j = (\underbrace{0, \cdots, 0}_{j-1}, \rho\eta_j/\rho\eta_j, \rho\eta_{j+1}/\rho\eta_j, \ldots, \rho\eta_m/\rho\eta_j)^t$$

Thus, the j^{th} column of $\hat{\bar{B}}$ is $v_j \cdot \sum_{\xi_{n+i} \in S_j} |\rho\eta_j/\rho\xi_{n+i}|$. From the form of v_j, it follows that $\hat{\bar{B}}$ is lower triangular. ■

A similar result for $\hat{\bar{A}}$, the condensed matrix for $\bar{A}$, does not follow. For binary trees, $n > r\text{-}n \geq m$, and seemingly the possibility exists for reducing the n columns of $\bar{A}$ to m columns of $\hat{\bar{A}}$. The matrix, $\hat{\bar{A}}$, then being lower triangular, would be of rank m, and this fact would guarantee that a backward analysis could be done. The argument below shows why this is not always true. The following example provides an illustration.



Here,

$$\bar{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \hat{\bar{A}} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \\ 2 & 2 \end{bmatrix}$$

In theorem 6.3, each set S_j, $1 \leq j \leq m$, is non-empty and hence, there is a condensed column for each set. For this example, S_3 is empty, characterizing the fact that each path from an input node to η_3 encounters another output node first. Thus, there is no condensed column corresponding to S_3, and $\hat{\bar{A}}$ has fewer than m columns. With the additional assumption that all the sets, S_j, $1 \leq j \leq m$, are non-empty, theorem 6.3 will hold for $\hat{\bar{A}}$.

6.3. Algorithms for g*

The preceding section has shown that the size of the problem for B-analysis (5.9) may be reduced to the following:

$$\begin{aligned} &\text{minimize } \Theta \\ &\text{subject to } \dot{\hat{A}}\hat{d} = \hat{\bar{B}}\hat{g}^* \\ &\qquad |\hat{\delta}_i| \leq \Theta,\ 1 \leq i \leq k_2+m, \end{aligned} \tag{6.6}$$

where $\hat{d}$ has components $\hat{\delta}_i$ representing, according to the reduction of $\dot{A}$, combined input and injected output relative errors, and $\hat{g}^*$ has components $\hat{\gamma}_i^*$, $1 \leq i \leq k_1$, representing, according to the reduction of $\bar{B}$, combined relative errors for the "worst" computation, with $|\hat{\gamma}_i^*| = 1$. Thus, algorithms for $\hat{g}^*$, rather than g*, will be presented.

The $\hat{g}^*$ which is sought is that $\hat{g}$ in C_{k_1} which gives the largest

solution to (6.6). The problem of finding $\hat{g}^*$ is a minimax problem. Unfortunately, algorithms for finding the global solution to this type of problem are often tedious. Because of the special form of $\hat{g}^*$, finite, though often lengthy, algorithms can be constructed to find the global solution to (6.6). Two such algorithms will be presented.

The first algorithm consists of solving (6.6) for all possible $\hat{g}$, and selecting that $\hat{g}$ which gives the largest Θ. Since each component of $\hat{g}^*$ is ± 1, and there are k_1 components, (6.6) must be solved 2^{k_1} times. The reason for the reduction of $\bar{B}$ to $\hat{\bar{B}}$ in the last section now becomes apparent.

For a given $\hat{g}$, the solution of (6.6) is a linear program. As it stands, (6.6) is not in standard form, since the variables $\hat{\dot{\delta}}$ are not constrained to be non-negative. A standard procedure for handling this case is to replace each inequality $|\hat{\dot{\delta}}_i| \leq \Theta$ by the two inequalities $\hat{\dot{\delta}}_i^+ - \hat{\dot{\delta}}_i^- \leq \Theta$, and $\hat{\dot{\delta}}_i^+ - \hat{\dot{\delta}}_i^- \geq -\Theta$, where $\hat{\dot{\delta}}_i^+$ and $\hat{\dot{\delta}}_i^-$ are non-negative variables representing the positive and negative parts of $\hat{\dot{\delta}}_i$. This procedure practically doubles the number of variables and constraints, and increases the storage required by a factor of four.

An alternative procedure consists of a change of variables. Define the non-negative variables $\hat{\dot{\delta}}_i' = \hat{\dot{\delta}}_i + \Theta$, i.e., $\hat{\dot{d}}' = \hat{\dot{d}} + \Theta \hat{e}_{k_2+m}$, where $\hat{e}_{k_2+m}$ is a k_2+m vector each of whose elements is 1. Then, the constraint $|\hat{\dot{\delta}}_i| \leq \Theta$, becomes $\hat{\dot{\delta}}_i' - 2\Theta \leq 0$, or $\hat{\dot{d}}' - 2\Theta \hat{e}_{k_2+m} \leq 0$. Substituting $\hat{\dot{d}} = \hat{\dot{d}}' - \Theta \hat{e}_{k_2+m}$, the constraint $\hat{\ddot{A}}\hat{d} = \hat{\bar{B}}\hat{g}^*$ becomes $\hat{\ddot{A}}\hat{d}' - \Theta w = \hat{\bar{B}}\hat{g}^*$, where w is a k_2+m vector having the elements $\omega_i = \sum_{j=1}^{k_2+m} \hat{\dot{a}}_{ij}$. Thus, (6.6) has become

$$\begin{aligned} &\text{minimize } \Theta \\ &\text{subject to } \hat{\dot{A}}\hat{\dot{d}}' - \Theta w = \hat{\bar{B}}\hat{g}^* \\ &\qquad\qquad \hat{\dot{d}}' - 2\Theta\hat{e}_{k_2+m} \leq 0 \end{aligned} \tag{6.7}$$

However, rather than solving (6.7), tremendous savings in time and storage can be had by considering a dual problem, Ascher [1]. His notation will be employed. Given a candidate for $\hat{g}^*$, let $\hat{\dot{d}}$ be a feasible solution to (6.6), i.e., $\hat{\dot{d}}$ satisfies $\hat{\dot{A}}\hat{\dot{d}} = \hat{\bar{B}}\hat{g}^*$. Define the following quantities: $\gamma = 1/\|\hat{\dot{d}}\|_\infty = 1/\Theta$, $K_0 = \|\hat{\dot{A}}\|_\infty / \|\hat{\bar{B}}\hat{g}^*\|_\infty$. Choose any $K > K_0$ and define the k_2+m+1 vector y with components $\eta_i = \gamma\,\hat{\dot{\delta}}_i$, $1 \leq i \leq k_2+m$, and $\eta_{k_2+m+1} = \gamma/K$. Since $\gamma \leq K_0$, $\|y\|_\infty \leq 1$. Let $\bar{A} = [\hat{\dot{A}}, -K\,\hat{\bar{B}}\hat{g}^*]$ be a $m\times(k_2+m+1)$ matrix, and c be a k_2+m+1 vector whose elements are zero except for the last component which is 1. Then, (6.6) is equivalent to the linear program:

$$\begin{aligned} &\max c^t y \\ &\text{subject to } \bar{A}y = 0 \\ &\qquad\qquad |\eta_i| \leq 1 \end{aligned} \tag{6.8}$$

The dual of (6.8), Rabinowitz [15], may be stated as

$$\begin{aligned} &\min \|r\|_1 \\ &\text{subject to } \bar{A}^t a + r = c \end{aligned} \tag{6.9}$$

where $\bar{A}$ and c are as described above. What is claimed to be the most efficient algorithm yet devised for solving (6.9) is provided by Barrodale and Roberts [2, 3]. The outstanding features of the algorithm include: (1) a condensed tableau, only $\bar{A}^t$ and c are explicitly stored; (2) no phase 1

i.e., an initial basic feasible solution is available, r = c; (3) the ability of passing through several vertices in a single iteration. The following equalities show how Θ is retrieved from the solution of (6.9).

$$\ell_1 = \min \|r\|_1 = \max c^t y = \eta_{k_2+m+1} = \gamma/K = 1/(\Theta K).$$

Hence, $\Theta = 1/(K\ell_1)$.

The following table shows the saving in time of (6.9) over (6.7) for a battery of problems and algorithms. The time, in seconds, is for the evaluation of Θ for one candidate for $\hat{g}^*$. It is assumed that $\hat{\dot{A}}$ and $\hat{\bar{B}}\hat{g}^*$ have previously been computed.

ALGORITHM	PRIMAL (6.7)	DUAL (6.9)
BACK4	0.23	0.02
SEMI4	0.23	0.02
PARA4	0.19	0.01
BACK8	3.91	0.10
SEMI8	3.77	0.09
PARA8	2.10	0.11
GS	1.12	0.15
MGS	1.00	0.15
LTBAND	1.82	0.20
RINVERSE	0.78	0.12
GJ	0.55	0.03
GE	0.65	0.03
CHOL	0.38	0.02
MATMUL	5.79	0.58
CG	0.15	0.01

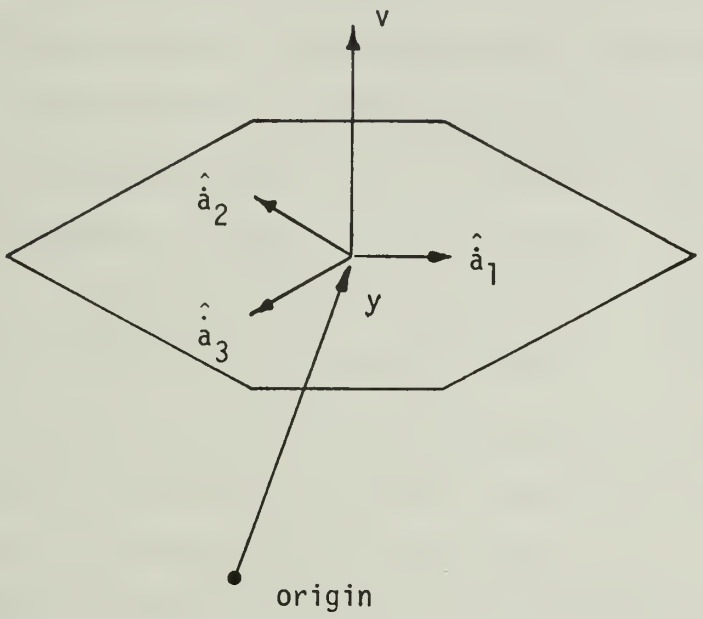
Table 6. Primal and Dual Solution Timings

While the first method for finding $\hat{g}^*$ searches all possible $\hat{g}$ in C_{k_1} for that $\hat{g}$ which gives the largest solution to (6.9), a second method searches all the faces of $\dot{A}\, C_{k_2+m}$ for that face which will cover $\hat{\bar{B}}\hat{g}^*$. The representation of $\dot{A}\, C_{k_2+m}$ is changed to a more standard representation which is in terms of the hyperplanes which support the polytope. A hyperplane, Householder [5], divides R^m into two half spaces and is represented by the equation $v^t x = \omega$, where v is an m-vector which is normal to the hyperplane, and ω is a constant which is related, by the length of v, to the Euclidean distance of the hyperplane from the origin. A hyperplane supports the polytope if there is at least one point in common between the two, and all the points of the polytope lie on the hyperplane or in only one of the half spaces. The points x in the polytope satisfy $v^t x \leq \omega$. Since the polytope is centrally symmetric, there is a corresponding support hyperplane $v^t x = -\omega$ and points in the polytope satisfy $v^t x \geq -\omega$. Collectively, the points in the polytope satisfy $|v^t x| \leq \omega$. The particular support hyperplanes of interest intersect the (m-1)-dimensional faces of the polytope.

The points on a face of the polytope $\dot{A}\, C_{k_2+m}$ have a particular form. Each vertex of the face is of the form $\dot{\hat{A}}\hat{d} = \sum_{i=1}^{k+m} \delta_i \dot{a}_i$ with $|\delta_i| = 1$. Points on the face are convex combinations of these vertices. A point on the face supported by a hyperplane with normal v may be represented by

$$\dot{\hat{A}}\hat{d} = \sum_{v^t \dot{a}_i \neq 0} \delta_i \dot{a}_i + \sum_{v^t \dot{a}_i = 0} \delta_i \dot{a}_i \qquad (6.10)$$

where the sum is split according to whether the projection of $\hat{\dot{a}}_j$ on v is zero. All of the points on the face have the same projection on v. The following illustration shows a typical two-dimensional face of an $\hat{\dot{A}}\ C_{k_2+m} \subseteq R^3$, where, in this case, three of the columns of $\hat{\dot{A}}$ are orthogonal to v,



and y is the first sum in (6.10). The points on the face are symmetric around the point y, which may be called the "center" of the face. The coefficients, $\hat{\dot{\delta}}_i$, for y are given by

$$\hat{\dot{\delta}}_i = \begin{cases} +1 & \text{if } v^t \hat{\dot{a}}_i > 0 \\ -1 & \text{if } v^t \hat{\dot{a}}_i < 0 \end{cases}$$

If this were not so, then it would be possible to construct a point, y', which would have a projection on v which is greater than that of the points on the face. The origin and y' would then lie in different half-spaces, contradicting the fact that the hyperplane supports the polytope on this

face. The coefficients, $\hat{\delta}_i$, in the second sum of (6.10) satisfy $|\hat{\delta}_i| \leq 1$ and are related to the representations of a point on the face in terms of convex combinations of the vertices of that face.

Since a face is (m-1)-dimensional, there must be at least m-1 terms in the second sum of (6.10). Each face therefore corresponds to a set of at least m-1 columns of $\hat{A}$. These vectors must span an (m-1)-dimensional subspace of R^m, and hence, attention is restricted to combinations of m-1 columns of $\hat{A}$ which are linearly independent. While there may be more than m-1 columns of $\hat{A}$ orthogonal to v, any subset of m-1 linearly independent vectors will suffice to determine the face and the support hyperplane.

Suppose m-1 linearly independent columns of $\hat{A}$ are given, say $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{m-1}$. Then the condition $v^t\hat{a}_i = 0$, $1 \leq i \leq m-1$, determines v to within a constant multiplier. Requiring that $\|v\|_2 = 1$ determines a specific v. Having found v, the corresponding ω is determined by finding a point which lies on the face. Such a point, y, has been determined above. The value of ω is then $|v^t y|$. Thus, for each set of m-1 linearly independent columns of $\hat{A}$, the corresponding symmetric pair of support hyperplanes can be determined. The points of the polytope must satisfy $|v^t x| \leq \omega$. If $k_3 \leq \binom{k_2+m}{m-1}$ is the number of support hyperplane pairs, then the polytope may be represented by those points x which satisfy $|Vx| \leq w$, where V is a $k_3 \times m$ matrix whose rows, v_j^t, are the normals for the support hyperplanes, w is a k_3-vector of corresponding ω's. Often $\binom{k_2+m}{m-1}$ is much less than $\binom{n+m}{m-1}$ and the reason for the reduction of $\dot{A}$ to $\hat{A}$ now becomes apparent. It will be convenient to work with a normalized version of these inequalities, $|\bar{V}x| \leq e$, where $e = (1,1,\ldots,1)^t$, in which each row has been divided by the corresponding right hand side.

The polytope $\hat{\bar{B}}\, C_{k_1}$ is covered by expanding $\dot{\hat{A}}\, C_{k_2+m}$ by a factor Θ. Corresponding to the expanded polytope $\dot{\hat{A}}\, \Theta C_{k_2+m}$ is the hyperplane representation

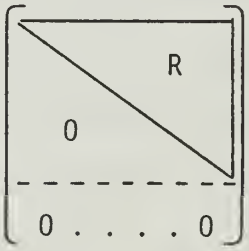
$$|\bar{V}x| \leq \Theta e \tag{6.11}$$

in which the support hyperplanes have been moved out away from the origin by a factor Θ. Given any x in R^m, the smallest Θ for which (6.11) holds is the norm of x, where here, instead of a unit ball or cube, there is a unit polytope. Of interest are the norms of the points which lie in $\hat{\bar{B}}\, C_{k_1}$. Such points are of the form $\hat{\bar{B}}\hat{g} = \sum_{i=1}^{k_1} \hat{\gamma}_i \hat{\bar{b}}_i$, with $|\hat{\gamma}_i| \leq 1$. The particular $x = \hat{\bar{B}}\hat{g}^*$ is sought which gives the largest Θ. It is known that this x is a vertex of $\hat{\bar{B}}\, C_{k_1}$, and $|\hat{\gamma}_i^*| = 1$, $1 \leq i \leq k_1$. Thus, the smallest Θ is sought for which <u>any</u> choice of $|\hat{\gamma}_i| \leq 1$ generates an $x = \hat{\bar{B}}\hat{g}$ which satisfies (6.11). Let $U = \bar{V}\,\hat{\bar{B}}$ be a $k_3 \times k_1$ matrix with elements μ_{ij}. Then, it must be that $|U\hat{g}| \leq \Theta e$. Let u_i^t be the i^{th} row of U. Then, since $|\hat{\gamma}_i| \leq 1$, $1 \leq i \leq k_1$, $|u_i^t \hat{g}| \leq \|u_i\|_1$, with equality when the components of $\hat{g}$ are chosen to have magnitude 1 and sign of the corresponding elements of u_i. Let J be that subscript for which $u_i^t \hat{g}_i$ is maximum, i.e., $u_J^t \hat{g}_J = \max_i u_i^t \hat{g}_i$, where $\hat{g}_i$ is that $\hat{g}$ chosen in the above manner for each row u_i^t. Then, $\hat{g}^* = \hat{g}_J$ with elements $\hat{\gamma}_i^* = 2 \cdot \text{sign}\,(\mu_{Ji}) - 1$, $1 \leq i \leq k_1$, and $\Theta = u_J^t \hat{g}^*$.

If $\dot{\hat{d}} \in \Theta C_{k_2+m}$ corresponding to $\hat{\bar{B}}\hat{g}^*$ is desired, additional computations need to be performed. Let $\bar{V}_J^t$ be the J^{th} row of $\bar{V}$, where $u_J^t = \bar{v}_J^t \hat{\bar{B}}$ and J is as defined above. Thus, $\bar{v}_J$ is the normal to the hyperplane which supports

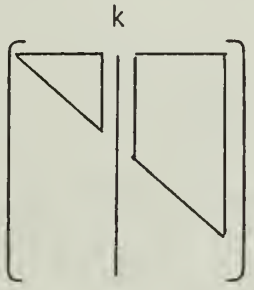
the face of $\dot{\hat{A}}\ \Theta C_{k_2+m}$ on which $\hat{\bar{B}}\hat{g}^*$ lies. Hence, $\hat{\bar{B}}\hat{g}^*$ may be represented as in (6.10). For the first sum in (6.10), $|\dot{\hat{\delta}}_i| = \pm\Theta$, where the sign is determined by the sign of the corresponding $\bar{v}_j^t \dot{\hat{a}}_i$. This follows from the fundamental theorem of linear programming which states that the inactive variables take on their extreme values. The values of $\dot{\hat{\delta}}_i$ in the second sum are determined by solving an m×(m-1) consistent system of equations involving those columns of $\dot{\hat{A}}$ for which $\bar{v}_j^t \dot{\hat{a}}_i = 0$, and the right hand side $(\hat{\bar{B}}\hat{g}^* - y)$. If the number of columns in $\dot{\hat{A}}$ for which $\bar{v}_j^t \dot{\hat{a}}_i \neq 0$ is greater than m-1, then the problem becomes a linear program.

In an implementation of this algorithm, the matrix U need not be stored. It may be computed one row at a time and a variable may hold the largest value of $u_i^t \hat{g}_i$ computed so far. Additionally, the normals, v_j, $1 \leq j \leq k_3$, may be easily computed in the following manner. Let A be an m×(m-1) matrix whose columns are some m-1 columns of $\dot{\hat{A}}$. Then, $A = Q\tilde{R}$, where $\tilde{R}$ has the form



where the last row of $\tilde{R}$ is zero. If rank (R) = m-1, then the m-1 columns of $\dot{\hat{A}}$ chosen are linearly independent and hence, determine an (m-1)-dimensional face. The normal to the hyperplane supporting this face is $v = Qe_m$, where e_m is the last column of the identity matrix of order m, and $\|v\|_2 = 1$.

When m is greater than 20, it is computationally advantageous to compute the QR decomposition and the normal v in a different manner. Rather than computing a new QR decomposition for each combination m-1 columns of $\hat{\hat{A}}$, the combinations may be ordered in such a way that the factorization of the next combination may be easily computed from that of the previous one. As before, let $A = Q\tilde{R}$, where A consists of some m-1 columns of $\hat{\hat{A}}$. Let A' be the same as A except that column k has been replaced by a column, $\hat{a}_j$, of $\hat{\hat{A}}$ so as to form a new combination. The revolving-door algorithm given by Nijenhuis, Wilf [13] specifies a sequence of replacements which will generate all $\binom{k_2+m}{m-1}$ combinations. Then $A' = Q\tilde{R}'$, where $\tilde{R}'$ has the form



and $\tilde{R}'$ is the same as $\tilde{R}$ except for the k^{th} column which is now $Q^t\hat{a}_j$. The matrix $\tilde{R}'$ may be reduced to the form of $\tilde{R}$ by premultiplication by an orthogonal matrix P consisting of a product of Givens plane rotations.

$$P = G'_{m-1} \cdots G'_{k+1} G_{k+1} \cdots G_m$$

where G_i introduces 0 in the i^{th} row of column k, and G'_i introduces 0 in the $i+1^{st}$ row of column i. The QR factorization of A' is then $A = (QP^t)(P\tilde{R}')$ or $A' = Q'R'$, and the normal vector corresponding to this new combination is $v' = Q'e_m$. This method has the disadvantages of requiring extra storage

for Q in an unfactored form, and of the loss of orthogonality in the columns of Q.

Both of the methods presented for finding $\hat{g}^*$ are exhaustive. Hence, these methods are viable only for small k_1 or k_3, due to the computation time involved. For larger problems, methods which obtain an approximation to Θ are required.

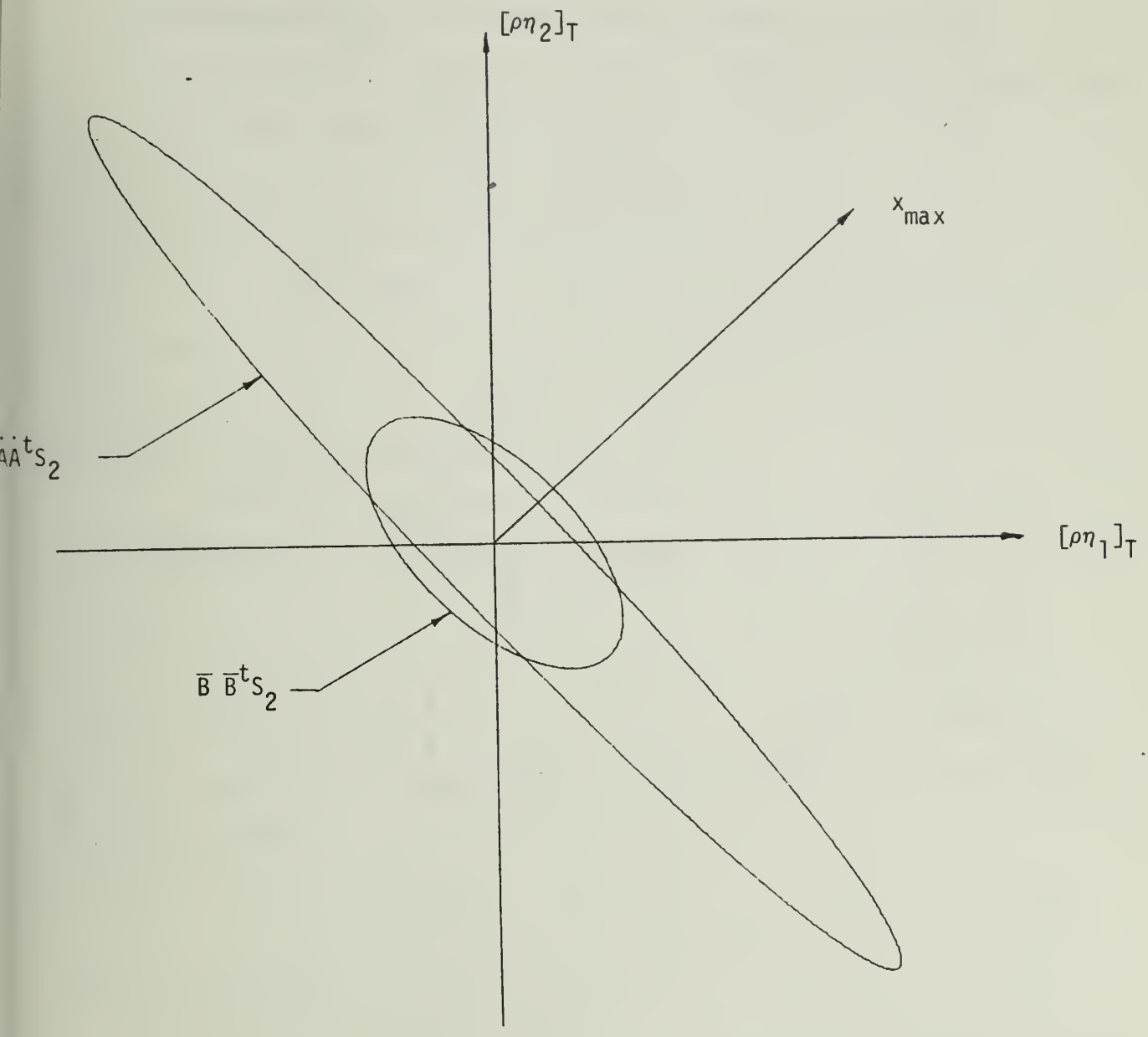
One algorithm to compute an approximation to g^* is based on the following heuristic. If the direction in which $\bar{B}g^*$ lies is known, then the choice of γ_i^* could be made according to whether $\bar{b}_i$, the i^{th} column of $\bar{B}$, or $-\bar{b}_i$ lies in that direction. The direction used is the eigenvector, x_{max}, corresponding to the largest eigenvalue, λ_{max}, of the generalized eigenvalue problem

$$\bar{B}\,\bar{B}^t x = \lambda \dot{A}\dot{A}^t x \tag{6.12}$$

This problem is similar to the problem which Miller [11] solves to obtain an approximation to Θ. He has shown that $(\lambda_{max})^{1/2}$ approximates Θ, and that his approximation satisfies

$$(n+m)^{-1/2}\,(\lambda_{max})^{1/2} \leq \Theta \leq (\lambda_{max})^{1/2}\,(r-n)^{1/2} \tag{6.13}$$

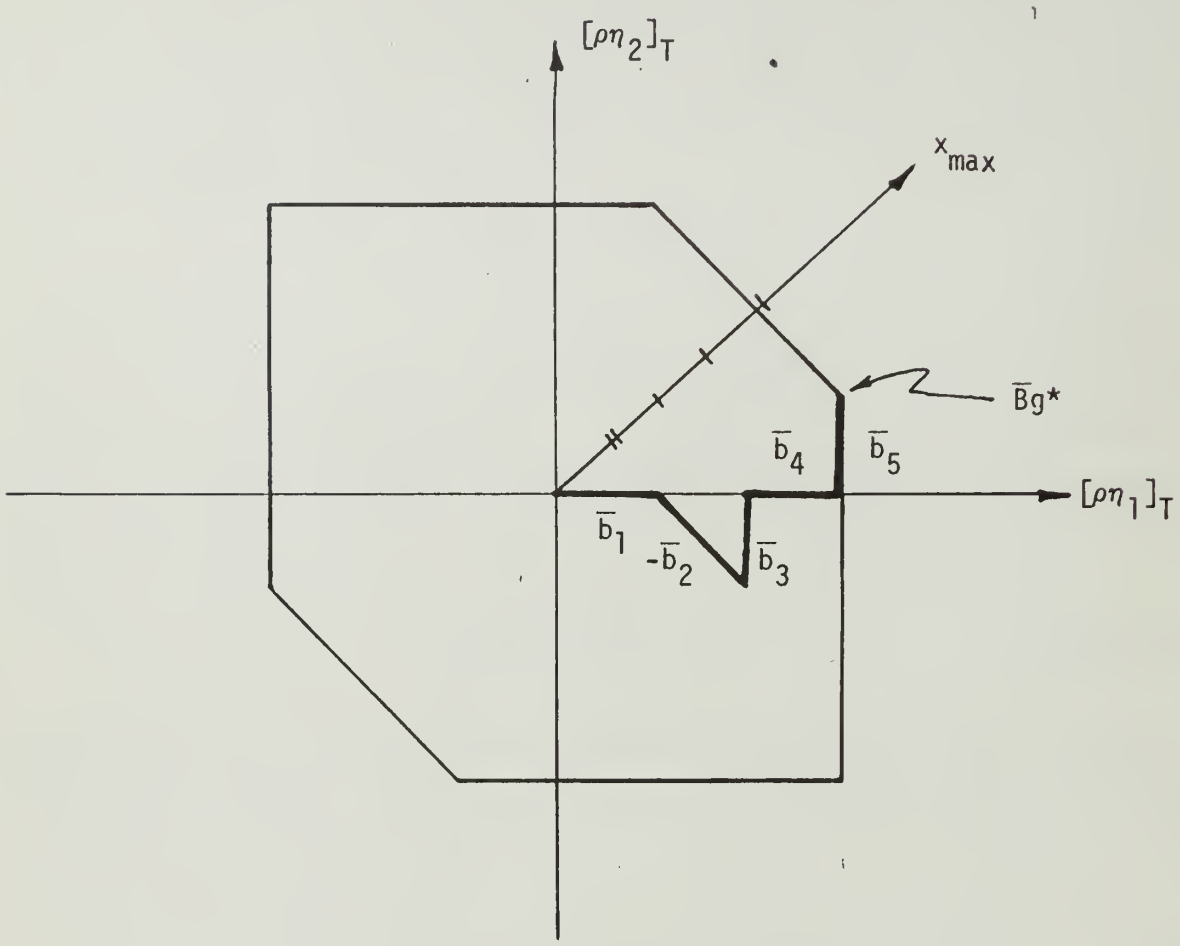
The following figure illustrates Miller's methods applied to the model problem.



The unit sphere $S_m = \{v \in R^m: \|v\|_2 = 1\}$ replaces the unit cubes C_{n+m} and C_{r-n}, and ellipsoids $\dot{A}\dot{A}^t S_m$ and $\bar{B}\,\bar{B}^t S_m$ replace the polytopes $\dot{A}\,C_{n+m}$ and $\bar{B}\,C_{r-n}$. The ellipsoid $\dot{A}\dot{A}^t S_m$ is expanded so as to just cover $\bar{B}\,\bar{B}^t S_m$. Solving (6.12) yields $(\lambda_{max})^{1/2} = 1.163$, and (6.13) states that Θ lies in the interval

$$0.4749 = (6)^{-1/2}\, 1.163 \leq \Theta \leq 1.1163\,(5)^{1/2} = 2.6$$

The following figure illustrates the use of x_{max} as a guide to approximating g* and hence $\bar{B}g^*$.



Each γ_i^*, $1 \leq i \leq r-n$, is chosen so that $\gamma_i^* \bar{b}_i$ has a positive projection on x_{max}.

$$\gamma_i^* = \begin{cases} +1 & \text{if } x_{max}^t \bar{b}_i \geq 0 \\ -1 & \text{if } x_{max}^t \bar{b}_i < 0 \end{cases} \tag{6.14}$$

For the model problem, (6.14) was successful in generating the true g*, and a dual linear program similar to (6.9) led to the calculation of $\Theta = 1.1176$. When applied to other problems, algorithm (6.14) did not always produce the true g*. Nevertheless, the resulting approximate Θ is a lower bound on the true Θ, since the approximate g* gives rise to a $\bar{B}g^*$ which is a vertex of $\bar{B}\ C_{r-n}$. Occasionally, the approximation to g* was improved by using (6.14) iteratively, the approximate $\bar{B}g^*$ obtained on the previous iteration serving as a guide for the present iteration. Frequently, the resulting increase in Θ was marginal.

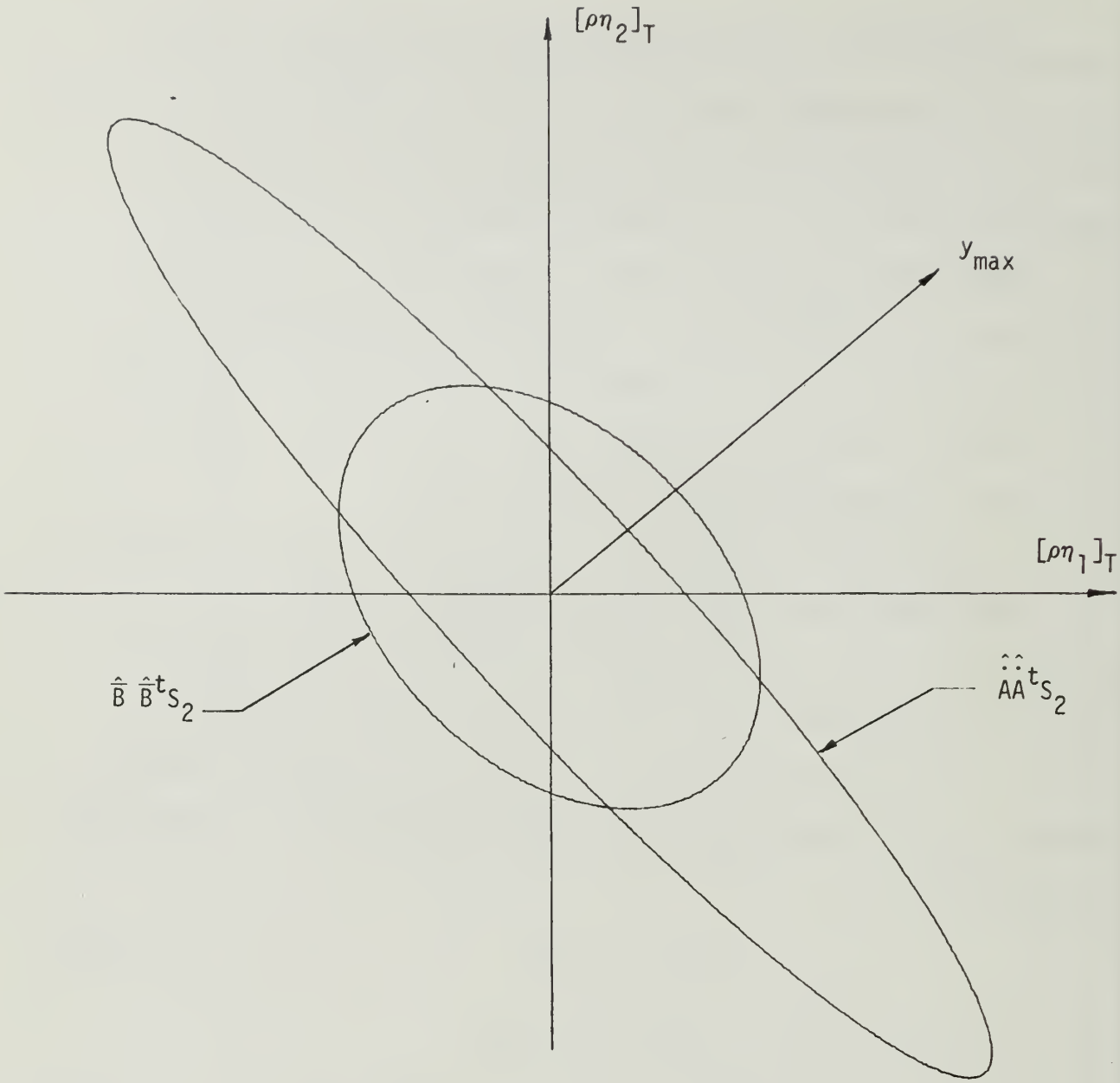
The preceding discussion has not made use of the simplification of polytope representation described in the previous section. Since $\bar{B}\ C_{r-n} = \hat{\bar{B}}\ C_{k_1}$ and $\dot{A}\ C_{n+m} = \hat{\dot{A}}\ C_{k_2+m}$, the same techniques used by Miller [11] in the development of (6.12) and (6.13) may be used to establish similar relationships based on $\hat{\bar{B}}\ C_{k_1}$ and $\hat{\dot{A}}\ C_{k_2+m}$. Thus, Θ may be approximated by $(\mu_{max})^{1/2}$, where μ_{max} is the largest eigenvalue of the generalized eigenvalue problem

$$\hat{\bar{B}}\ \hat{\bar{B}}^t y = \mu \hat{\dot{A}} \hat{\dot{A}}^t y \tag{6.15}$$

Also, the approximation, $(\mu_{max})^{1/2}$, satisfies

$$(k_2+m)^{-1/2}\ (\mu_{max})^{1/2} \leq \Theta \leq (\mu_{max})^{1/2}\ (k_1)^{1/2} \tag{6.16}$$

The following figure illustrates the ellipses in (6.15) for the model problem.



Solving (6.15) yields $(\mu_{max})^{1/2} = 1.270$, and (6.16) states that Θ lies in the interval.

$$0.6342 = (4)^{-1/2}\, 1.270 \leq \Theta \leq 1.270\,(3)^{1/2} = 2.200$$

This interval is smaller than that provided by Miller, and similar results were found for most of examples tested. In all cases, the lower bound

was larger than Miller's, though a proof that this is always true is lacking. Additionally, it was observed that $(\mu_{max})^{1/2} > (\lambda_{max})^{1/2}$ and that, in most cases, $(\mu_{max})^{1/2}$ was a much better approximation to Θ than $(\lambda_{max})^{1/2}$. Table 7 provides examples for comparison of (6.12) and (6.15).

An algorithm similar to (6.14) based on the eigenvector guide, y_{max}, corresponding to μ_{max}, performed neither consistently better nor consistently worse. The values of $\hat{\gamma}_i^*$ are chosen in the following manner.

$$\hat{\gamma}_i^* = \begin{cases} +1 & \text{if } y_{max}^t \hat{\bar{b}}_i \geq 0 \\ -1 & \text{if } y_{max}^t \hat{\bar{b}}_i < 0 \end{cases} \qquad (6.17)$$

The resulting approximate $\hat{\bar{B}}\hat{g}^*$ is used in (6.9). Table 8 provides examples for comparison. The true values of Θ are computed by one of the exhaustive methods.

ALGORITHM	true Θ	(6.12) $(\lambda_{max})^{1/2}$	(6.13) Interval for Θ	(6.15) $(\mu_{max})^{1/2}$	(6.16) Interval for Θ
BACK4	1.5879	1.390	[.3855, 4.815]	1.981	[.8086, 3.430]
SEMI4	1.2345	1.166	[.3234, 4.039]	1.526	[.6229, 2.643]
PARA4	1.9547	1.380	[.3827, 5.163]	2.126	[.8678, 4.251]
BACK8	2.4139	1.750	[.2669, 13.09]	2.823	[.7544, 7.468]
SEMI8	1.7148	1.426	[.2175, 10.67]	1.998	[.5340, 5.286]
PARA8	7.6503	2.179	[.3323, 20.21]	5.1468	[1.376, 17.07]
GS	10.723	3.488	[.7120, 30.39]	6.841	[1.569, 34.88]
MGS	11.032	3.408	[.6956, 31.67]	6.748	[1.548, 35.71]
LTBAND	4.2329	2.098	[.3449, 19.46]	3.985	[1.029, 12.60]
RINVERSE	18,329.	(2)	-	23,936.	[5,491.,75,691.]
GJ	2.3719	1.341	[.2737, 11.69]	2.438	[.8621, 6.451]
GE	2.1406	1.273	[.2599, 10.02]	2.623	[.9273, 6.424]
CHOL	2.3450	1.395	[.3288, 10.98]	2.747	[.7343, 9.906]
MATMUL	(1)	2.723	[.3930, 28.82]	6.404	[.9243, 25.62]
CG	6.3204	3.150	[.9093, 34.79]	5.2403	[1.747, 29.65]

Notes: (1) Computational time was prohibitive for exhaustive methods.
(2) Miller's version of $\overline{B}\,\overline{B}^t$ in (6.12) was not positive-definite.

Table 7. Approximations to Θ

ALGORITHM	true Θ	(6.14) $\Theta(x_{max})$	(6.17) $\Theta(y_{max})$
BACK4	1.5879	1.5879	1.5879
SEMI4	1.2345	1.2345	1.2345
PARA4	1.9547	1.9547	1.9547
BACK8	2.4139	2.3014	2.3014
SEMI8	1.7148	1.6501	1.6501
PARA8	7.6503	5.1575	5.1575
GS	10.723	7.5674	7.5674
MGS	11.032	7.0978	6.685
LTBAND	4.2329	4.1062	4.1063
RINVERSE	18,329.	15,314.	15,314.
GJ	2.3719	2.3719	2.3719
GE	2.1406	2.1406	2.1223
CHOL	2.3450	2.3393	2.1699
MATMUL	-	5.7605	5.8892
CG	6.3204	5.7967	5.7967

Table 8. Eigenvector Guide Approximations

7. Comparison of Algorithms

There are often several algorithms which may be used to solve a given problem, or set of similar problems. The question arises as to which of the competing algorithms should have been used. The answer will depend upon the criteria used for selection. The analyses of the previous sections provide several alternative criteria. Attention will be restricted to a choice between two algorithms.

The simplest measurement of competing algorithms is the forward analysis. One algorithm is preferable to another if it can guarantee a smaller bound on the relative error in the solution. Let $\overline{B}_i$ be the matrix of partial relative derivatives of outputs with respect to computations for algorithm i. Then, algorithm 1 is preferable to algorithm 2 if

$$\|\overline{B}_1\|_\infty < \|\overline{B}_2\|_\infty \tag{7.1}$$

A refinement of (7.1) may use (5.6) to compare the relative error bounds for each output individually.

Miller [8] proposes a measurement based on the polytopes $\overline{B}_1\ C_{r_1-n}$ and $\overline{B}_2\ C_{r_2-n}$, where r_i-n is the number of computations in algorithm i, and C_k is the unit cube in R^k. He defines the numbers

$$\begin{aligned} \omega_{1/2} &= \text{glb}\{\omega \geq 0: \overline{B}_1\ C_{r_1-n} \subseteq \omega\, \overline{B}_2\ C_{r_2-n}\} \\ \omega_{2/1} &= \text{glb}\{\omega \geq 0: \overline{B}_2\ C_{r_2-n} \subseteq \omega\, \overline{B}_1\ C_{r_1-n}\} \end{aligned} \tag{7.2}$$

These numbers may be interpreted as follows. If algorithm 1 produces a computed result with local relative errors bounded by ϵ, then the same computed result can be obtained by algorithm 2 with local relative errors

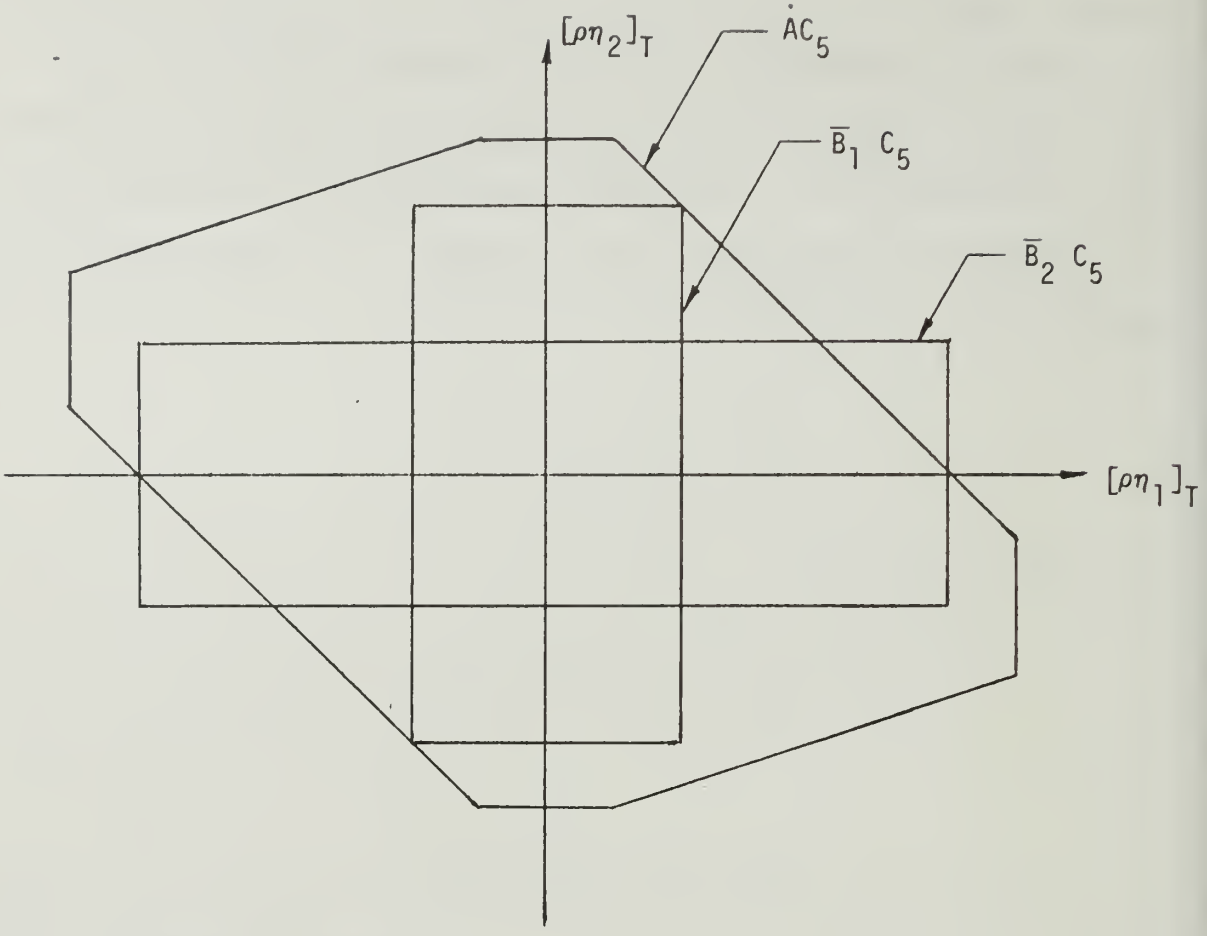
bounded by $\omega_{1/2} \cdot \epsilon$. An analogous interpretation holds for $\omega_{2/1}$. Clearly, algorithm 1 is preferable to algorithm 2 if $\omega_{1/2} > 1$, and algorithm 2 is more attractive if $\omega_{2/1} > 1$. However, as Miller shows, $\omega_{1/2} \cdot \omega_{2/1} \geq 1$, and it is possible for both $\omega_{1/2}$ and $\omega_{2/1}$ to be greater than 1 indicating that in some respect each algorithm is to be preferred to the other. This possibility poses a dilemma in the choice between two algorithms. Consider the following problem:

$$\eta_1 = \xi_1 \times (\xi_2 + \xi_3)$$
$$\eta_2 = \xi_3 \times (\xi_1 + \xi_2)$$

and the two algorithms:

$$1: \quad \eta_1 = \xi_1 \times (\xi_2 + \xi_3)$$
$$\eta_2 = \xi_3 \times \xi_1 + \xi_3 \times \xi_2$$
$$2: \quad \eta_1 = \xi_1 \times \xi_2 + \xi_1 \times \xi_3$$
$$\eta_2 = \xi_3 \times (\xi_1 + \xi_2)$$

For the data $\xi_1 = -1$, $\xi_2 = 2$, $\xi_3 = -3$, the polytopes $\dot{A}C_5$, $\overline{B}_1C_5$, and $\overline{B}_2C_5$ are shown below. Using (7.2) to compare these two algorithms gives $\omega_{1/2} = 3$, and $\omega_{2/1} = 2$. These results supply conflicting information for choosing between the two algorithms.



This dilemma may be overcome by choosing the polytope of the problem as the standard against which the polytopes of the algorithms are compared. Define the following numbers:

$$\begin{aligned}
\Theta_1 &= \text{glb}\{\Theta \geq 0: \bar{B}_1\ C_{r_1-n} \subseteq \Theta\dot{A}C_{n+m}\} \\
\Theta_2 &= \text{glb}\{\Theta \geq 0: \bar{B}_2\ C_{r_2-n} \subseteq \Theta\dot{A}C_{n+m}\} \\
\Theta_{1/2} &= \Theta_1/\Theta_2 \\
\Theta_{2/1} &= \Theta_2/\Theta_1
\end{aligned} \tag{7.3}$$

The numbers $\Theta_{1/2}$ and $\Theta_{2/1}$ compare the algorithms by taking ratios of the

stability numbers determined by a B-analysis. Clearly, algorithm 1 is more stable if $\Theta_{1/2} < 1$, and algorithm 2 is more stable if $\Theta_{2/1} < 1$. From the definitions, it follows immediately that $\Theta_{1/2} \cdot \Theta_{2/1} = 1$, implying that each algorithm cannot be more stable than the other. Applying (7.3) to the two algorithms gives $\Theta_1 = 1$, $\Theta_2 = 4/3$, $\Theta_{1/2} = 3/4$, and $\Theta_{2/1} = 4/3$, indicating that algorithm 1 is the preferred algorithm.

Bounds may be derived which relate the numbers defined in (7.2) and (7.3). These bounds make use of the following definitions:

$$\begin{aligned} \bar{\Theta}_1 &= \text{glb}\{\Theta \geq 0: \dot{A}C_{n+m} \subseteq \Theta\bar{B}_1\, C_{r_1-n}\} \\ \bar{\Theta}_2 &= \text{glb}\{\Theta \geq 0: \dot{A}C_{n+m} \subseteq \Theta\bar{B}_2\, C_{r_2-n}\} \end{aligned} \tag{7.4}$$

From (7.3),

$$\bar{B}_1\, C_{r_1-n} \subseteq \Theta_1\, \dot{A}C_{n+m}$$

Using (7.4),

$$\bar{B}_1\, C_{r_1-n} \subseteq \Theta_1\, \bar{\Theta}_2\, \bar{B}_2\, C_{r_2-n}$$

But (7.2) states

$$\bar{B}_1\, C_{r_1-n} \subseteq \omega_{1/2}\, \bar{B}_2\, C_{r_2-n}$$

from which it follows that

$$\Theta_1\, \bar{\Theta}_2 \geq \omega_{1/2}$$

Multiplying the left hand side by 1 in the form Θ_2/Θ_2 results in

$$\Theta_2 \bar{\Theta}_2 \Theta_{1/2} \geq \omega_{1/2}$$

or

$$\Theta_{1/2} \geq \omega_{1/2}/(\Theta_2 \bar{\Theta}_2) \tag{7.5}$$

Similar steps would show that

$$\Theta_{2/1} \geq \omega_{2/1}/(\Theta_1 \bar{\Theta}_1) \tag{7.6}$$

The quantities $(\Theta_i \bar{\Theta}_i) \geq 1$, $i = 1, 2$, may be called similarity coefficients. They measure the similarity of the polytopes $\dot{A}C_{n+m}$ and $\bar{B}_i C_{r_i-n}$, and have value 1 when the polytopes are the same.

From (7.2),

$$\bar{B}_1 C_{r_1-n} \subseteq \omega_{1/2} \bar{B}_2 C_{r_2-n}$$

Using (7.3)

$$B_1 C_{r_1-n} \subseteq \omega_{1/2} \Theta_2 \dot{A}C_{n+m}$$

But (7.3) states

$$\bar{B}_1 C_{r_1-n} \subseteq \Theta_1 \dot{A}C_{n+m}$$

from which it follows that

$$\omega_{1/2} \Theta_2 \geq \Theta_1$$

or

$$\omega_{1/2} \geq \Theta_{1/2} \tag{7.7}$$

Similar steps would show that

$$\omega_{2/1} \geq \Theta_{2/1} \tag{7.8}$$

Collectively, (7.5) and (7.7) give

$$\omega_{1/2} \geq \Theta_{1/2} \geq \omega_{1/2}/(\Theta_2 \bar{\Theta}_2)$$

and (7.6) and (7.8) produce

$$\omega_{2/1} \geq \Theta_{2/1} \geq \omega_{2/1}/(\Theta_1 \bar{\Theta}_1)$$

Using (7.3), a variety of algorithms for solving dense systems were compared for various data. The results are given in the following tables.

Algorithm 2

$\Theta_{1/2}$ Algorithm 1	G.E. w/o pivoting	G.J.	Givens	House	G.S.	M.G.S.
G.E. w/o pivoting	1.0	.549	1.14×10^{3}	5.56×10^{2}	5.21×10^{2}	7.46×10^{2}
G.J.	1.82	1.0	2.07×10^{3}	1.01×10^{3}	9.52×10^{2}	1.36×10^{3}
Givens	$.879\times10^{-3}$	$.483\times10^{-3}$	1.0	.490	.459	.654
House	$.180\times10^{-2}$	$.988\times10^{-3}$	2.04	1.0	.935	1.34
G.S.	$.192\times10^{-2}$	$.105\times10^{-2}$	2.18	1.07	1.0	1.43
M.G.S.	$.134\times10^{-2}$	$.737\times10^{-3}$	1.53	.747	.701	1.0

Data: well-conditioned, Miller [12]

$$A = \begin{bmatrix} 2.908 & -2.253 & 6.775 & 3.970 \\ 1.212 & 1.995 & 2.266 & 8.008 \\ 4.552 & 5.681 & 8.850 & 1.302 \\ -5.809 & -5.030 & 0.099 & 7.832 \end{bmatrix}, \quad b = \begin{bmatrix} 6.291 \\ 7.219 \\ 5.730 \\ 9.574 \end{bmatrix}, \quad x = \begin{bmatrix} -.7238 \\ .1108 \\ .8389 \\ .7460 \end{bmatrix}$$

Algorithm 2

Algorithm 1	$\Theta_{1/2}$ G.E. w/o pivoting	G.J.	Givens	House	G.S.	M.G.S.
G.E. w/o pivoting	1.0	.980	.571	.328	$.117\times10^{-3}$	$.385\times10^{-3}$
G.J.	1.02	1.0	.588	.333	$.120\times10^{-3}$	$.394\times10^{-3}$
Givens	1.75	1.70	1.0	.588	$.204\times10^{-3}$	$.671\times10^{-3}$
House	3.05	3.00	1.70	1.0	$.357\times10^{-3}$	$.117\times10^{-2}$
G.S.	8.55×10^{3}	8.35×10^{3}	4.90×10^{3}	2.80×10^{3}	1.0	3.33
M.G.S.	2.60×10^{3}	2.54×10^{3}	1.49×10^{3}	8.52×10^{2}	.300	1.0

Data: ill-conditioned - Vandermonde

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1/2 & 1/4 & 1/8 & 1/16 \\ 1/4 & 1/16 & 1/64 & 1/256 \\ 1/8 & 1/64 & 1/512 & 1/4096 \end{bmatrix}, \quad b = \begin{bmatrix} 10 \\ 13/8 \\ 7/16 \\ 167/1024 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Algorithm 2

Algorithm 1	$\Theta_{1/2}$ G.E. w/o pivoting	G.J.	Givens	House	G.S.	M.G.S.
G.E. w/o pivoting	1.0	.441	.521	.174	$.613\times10^{-1}$	$.714\times10^{-1}$
G.J.	2.27	1.0	1.18	.395	.139	.162
Givens	1.92	.846	1.0	.333	.118	.137
House	5.76	2.53	3.00	1.0	.352	.410
G.S.	1.63×10^{1}	7.19	8.50	2.84	1.0	1.17
M.G.S.	1.40×10^{1}	6.17	7.30	2.44	.858	1.0

Data: symmetric

$$A = \begin{bmatrix} 6 & 4 & 4 & 1 \\ 4 & 6 & 1 & 4 \\ 4 & 1 & 6 & 4 \\ 1 & 4 & 4 & 6 \end{bmatrix}, \quad b = \begin{bmatrix} 30 \\ 35 \\ 40 \\ 45 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Algorithm 2

$\Theta_{1/2}$ Algorithm 1	G.E. w/o pivoting	G.J.	Givens	House	G.S.	M.G.S.
G.E. w/o pivoting	1.0	.917	.223	.145	.221	.213
G.J.	1.09	1.0	.244	.159	.242	.233
Givens	4.48	4.10	1.0	.654	.990	.952
House	6.88	6.30	1.53	1.0	1.52	1.47
G.S.	4.53	4.14	1.01	.658	1.0	.962
M.G.S.	4.69	4.29	1.05	.682	1.04	1.0

Data: diagonally dominant

$$A = \begin{bmatrix} 4 & 1 & 1 & 1 \\ 2 & -5 & 1 & 1 \\ 1 & 2 & 6 & 1 \\ 3 & 1 & 2 & -7 \end{bmatrix}, \quad b = \begin{bmatrix} 13 \\ -1 \\ 27 \\ -17 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Algorithm 2

$\Theta_{1/2}$ Algorithm 1	G.E. w/o pivoting	G.J.	Givens	House	G.S.	M.G.S.
G.E. w/o pivoting	1.0	.935	.210	.104	.220	.198
G.J.	1.07	1.0	.225	.111	.235	.211
Givens	4.76	4.45	1.0	.493	1.05	.943
House	9.65	9.02	2.03	1.0	2.12	1.91
G.S.	4.55	4.25	.956	.471	1.0	.901
M.G.S.	5.06	4.73	1.06	.524	1.11	1.0

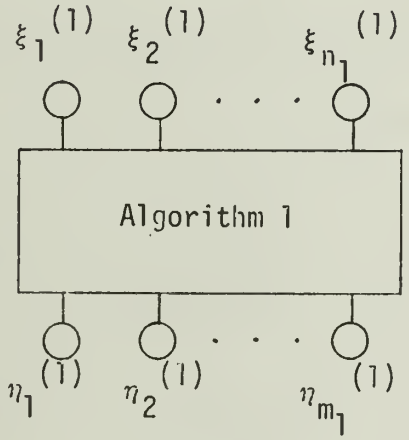
Data: positive definite

$$A = \begin{bmatrix} 10 & 3 & 2 & 1 \\ 3 & 11 & 3 & 2 \\ 2 & 3 & 12 & 3 \\ 1 & 2 & 3 & 13 \end{bmatrix}, \quad b = \begin{bmatrix} 26 \\ 42 \\ 56 \\ 66 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

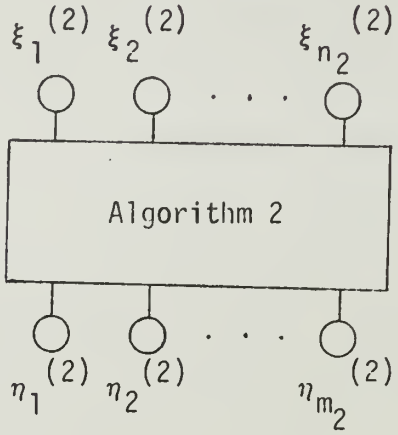
8. Composite Algorithms

A composite algorithm is one which is made up of smaller sub-algorithms where the outputs of one sub-algorithm serve as the inputs to the next. Often an algorithm can be thought of as composite by partitioning it into parts which perform distinct logical functions. For example, an algorithm for solving a system of equations may be considered a concatenation of two algorithms, one of which decomposes the matrix of coefficients followed by an algorithm which performs substitutions. The purpose of considering algorithms as composite is to be able to analyze an algorithm in terms of analyses done on its parts. Additionally, time may be saved in analyzing several algorithms by analyzing only once their common sub-algorithm.

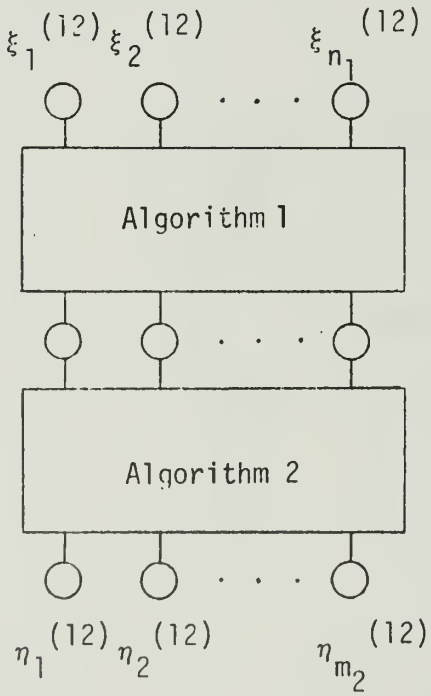
Without loss of generality, a composite Algorithm 12 will be considered as Algorithm 1 followed by Algorithm 2. Let Algorithm 1 have n_1 inputs, $\xi_i^{(1)}$, $1 \leq i \leq n_1$; m_1 outputs, $\eta_i^{(1)}$, $1 \leq i \leq m_1$; and $r_1 - n_1$ computational nodes, $\xi_i^{(1)}$, $n_1 < i \leq r_1$.



Similarly, let Algorithm 2 have n_2 inputs $\xi_i^{(2)}$, $1 \leq i \leq n_2$; m_2 outputs $\eta_i^{(2)}$, $1 \leq i \leq m_2$; and r_2-n_2 computational nodes, $\xi_i^{(2)}$, $n_2 < i \leq r_2$.



If $m_1 = n_2$, then the outputs of Algorithm 1 may be equated with the inputs of Algorithm 2, and the composite Algorithm 12 may be formed.



Algorithm 12 has n_1 inputs, $\xi_i^{(12)} = \xi_i^{(1)}$, $1 \le i \le n_1$; m_2 outputs, $\eta_i^{(12)} = \eta_i^{(2)}$, $1 \le i \le m_2$; and $r_1-n_1 + r_2-n_2$ computational nodes,

$$\xi_i^{(12)} = \begin{cases} \xi_i^{(1)}, & \text{if } i \le r_1 \\ & n_1 < i \le r_1+r_2-n_2 \\ \xi_{i-r_1+n_2}^{(2)}, & \text{if } i > r_1 \end{cases}$$

Associated with Algorithms 1, 2 and 12 are the matrices of partial relative derivatives of outputs with respect to input and computational nodes. For Algorithm 1, $\bar{A}_1$ is an $m_1 \times n_1$ matrix with elements

$$\bar{a}_{ij}^{(1)} = \rho\eta_i^{(1)}/\rho\xi_j^{(1)}$$

and $\bar{B}_1$ is an $m_1 \times (r_1-n_1)$ matrix with elements

$$\bar{\beta}_{ij}^{(1)} = \rho\eta_i^{(1)}/\rho\xi_{n_1+j}^{(1)}$$

Similarly, for Algorithm 2, $\bar{A}_2$ is an $m_2 \times n_2$ matrix with elements

$$\bar{a}_{ij}^{(2)} = \rho\eta_i^{(2)}/\rho\xi_j^{(2)}$$

and $\bar{B}_2$ is an $m_2 \times (r_2-n_2)$ matrix with elements

$$\bar{a}_{ij}^{(2)} = \rho\eta_i^{(2)}/\rho\xi_{n_2+j}^{(2)}$$

Finally, for Algorithm 12, $\bar{A}_{12}$ is an $m_2 \times n_1$ matrix with elements

$$\bar{a}_{ij}^{(12)} = \rho\eta_i^{(12)}/\rho\xi_j^{(12)} = \rho\eta_i^{(2)}/\rho\xi_j^{(1)}$$

and $\bar{B}_{12}$ is an $m_2 \times (r_1-n_1 + r_2-n_2)$ matrix with elements

$$\bar{\beta}_{ij}^{(12)} = \rho\eta_i^{(12)}/\rho\xi_{n_1+j}^{(12)} = \begin{cases} \rho\eta_i^{(2)}/\rho\xi_{n_1+j}^{(1)}, & \text{if } j \leq r_1-n_1 \\ \rho\eta_i^{(2)}/\rho\xi_{j-r_1+n_1+n_2}^{(2)}, & \text{if } j > r_1-n_1 \end{cases}$$

The matrices for Algorithm 12 may be computed from the matrices for Algorithms 1 and 2. Consider an element of $\bar{A}_{12}$,

$$\bar{a}_{ij}^{(12)} = \rho\eta_i^{(2)}/\rho\xi_j^{(1)}$$

Any path between $\eta_i^{(2)}$ and $\xi_j^{(1)}$ must pass through $\eta_k^{(1)} = \xi_k^{(2)}$, $1 \leq k \leq n_2$, the outputs of Algorithm 1 and inputs of Algorithm 2. Hence, by the path product property of partial relative derivatives,

$$\begin{aligned} \bar{a}_{ij}^{(12)} &= \sum_{k=1}^{n_2} (\rho\eta_i^{(2)}/\rho\xi_k^{(2)})(\rho\eta_k^{(1)}/\rho\xi_j^{(1)}) \\ &= \sum_{k=1}^{n_2} \bar{a}_{ik}^{(2)} \bar{a}_{kj}^{(1)} \end{aligned}$$

Thus,

$$\bar{A}_{12} = \bar{A}_2 \cdot \bar{A}_1 \tag{8.1}$$

Let $1 \leq j \leq r_1-n_1$, and consider an element of $\bar{B}_{12}$,

$$\bar{\beta}_{ij}^{(12)} = \rho\eta_i^{(2)}/\rho\xi_{n_1+j}^{(1)}$$

By employing the path product argument,

$$\begin{aligned} \bar{\beta}_{ij}^{(12)} &= \sum_{k=1}^{n_2} (\rho\eta_i^{(2)}/\rho\xi_k^{(2)})(\rho\eta_k^{(1)}/\rho\xi_{n_1+j}^{(1)}) \\ &= \sum_{k=1}^{n_2} \bar{a}_{ik}^{(2)} \bar{\beta}_{kj}^{(1)} \end{aligned}$$

For $1 \le j \le r_2-n_2$,

$$\bar{\beta}^{(12)}_{i,r_1-n_1+j} = \rho\eta_i^{(2)}/\rho\xi^{(2)}_{n_2+j} = \bar{\beta}^{(2)}_{ij}$$

Thus

$$\bar{B}_{12} = (\bar{A}_2\,\bar{B}_1,\ \bar{B}_2) \tag{8.2}$$

Using (8.1) and (8.2) to compute $\bar{A}_{12}$ and $\bar{B}_{12}$ from $\bar{A}_1$, $\bar{B}_1$, $\bar{A}_2$, and $\bar{B}_2$ may be more efficient than computing them by (4.1) from $\tilde{A}_{12}$ and $\tilde{B}_{12}$, where $\tilde{A}_{12}$ is an $(r_1-n_1 + r_2-n_2)\times n_1$ matrix of the form

$$\tilde{A}^t_{12} = (\tilde{A}_1{}^t,\ 0)$$

and $\tilde{B}_{12}$ is an $(r_1-n_1 + r_2-n_2)\times(r_1-n_1 + r_2-n_2)$ matrix of the form

$$\tilde{B}_{12} = \begin{bmatrix} \tilde{B}_1 & 0 \\ X & \tilde{B}_2 \end{bmatrix}$$

where X is zero except that column $\omega_i^{(1)}$ contains column i of $\tilde{A}_2$, where $\omega_i^{(1)}$ is the variable which indexes the outputs in Algorithm 1, i.e., if $k = \omega_i^{(1)}$, then $\eta_i^{(1)} = \xi_k^{(1)}$, $1 \le i \le n_2$. The matrices $\tilde{A}_1$, $\tilde{A}_2$, $\tilde{B}_1$, and $\tilde{B}_2$ are those defined in (3.7) for Algorithms 1 and 2. The number of operations required by (4.1) is at most $4m_2(r_1-n_1 + r_2-n_2)$ while (8.1) and (8.2) require $2m_2r_1m_1$ operations (note the independence from r_2). Thus, if r_2 is greater than $r_1(m_1-2)/2+n_1+n_2$, then $\bar{A}_{12}$ and $\bar{B}_{12}$ are more efficiently computed by (8.1) and (8.2).

Formulas (8.1) and (8.2) may be used to perform a forward analysis of Algorithm 12 in terms of forward analyses of Algorithms 1 and 2.

From (5.4),

$$|[\rho\eta_i^{(12)}]_T| \leq \sum_{j=1}^{n_1} |\rho\eta_i^{(12)}/\rho\xi_j^{(12)}| + \sum_{j=n_1+1}^{r_1+r_2-n_2} |\rho\xi_i^{(12)}/\rho\xi_j^{(12)}|$$

$$\leq \sum_{j=1}^{n_1} |\bar{\alpha}_{ij}^{(12)}| + \sum_{j=n_1+1}^{r_1+r_2-n_2} |\bar{\beta}_{i,j-n_1}^{(12)}| \tag{8.3}$$

$$\leq \|\bar{A}_{12}\|_\infty + \|\bar{B}_{12}\|_\infty \tag{8.4}$$

where the same bound is chosen for all i, $1 \leq i \leq m_2$. Using (8.1), the condition number of the problem may be bounded by

$$\|\bar{A}_{12}\|_\infty \leq \|\bar{A}_2\|_\infty \cdot \|\bar{A}_1\|_\infty$$

and using (8.2), the condition number of the algorithm may be bounded by

$$\|\bar{B}_{12}\|_\infty \leq \|\bar{A}_2\|_\infty \cdot \|\bar{B}_1\|_\infty + \|\bar{B}_2\|_\infty$$

Naturally, these easily computed bounds will be more pessimistic than those of (8.4), or (8.3) which require the calculation of $\bar{A}_{12}$ and $\bar{B}_{12}$.

A backward analysis for Algorithm 12 can be done if a backward analysis is possible for both Algorithms 1 and 2. The rank $(\bar{A}_{12})$ will be m_2, if rank $(\bar{A}_1) = m_1 \geq m_2$, and rank $(\bar{A}_2) = m_2$. Let the stability numbers for backward analysis for Algorithm 1 and 2 be denoted

$$\Theta_1 = \text{glb}\{\Theta \geq 0: \bar{B}_1 C_{r_1-n_1} \subseteq \Theta\bar{A}_1 C_{n_1}\} \tag{8.5}$$

and

$$\Theta_2 = \text{glb}\{\Theta \geq 0: \bar{B}_2 C_{r_2-n_2} \subseteq \Theta\bar{A}_2 C_{n_2}\} \tag{8.6}$$

respectively.

From (8.5) it follows that

$$\bar{A}_2 \bar{B}_1 \bar{C}_{r_1-n_1} \subseteq \Theta_1 \bar{A}_2 \bar{A}_1 \bar{C}_{n_1} \tag{8.7}$$

If Θ' is defined by

$$\Theta' = \text{glb}\{\Theta \geq 0: C_{n_2} \subseteq \Theta \bar{A}_1 C_{n_1}\} \tag{8.8}$$

(such as Θ' exists since rank $(\bar{A}_1) = m_1 = n_2$) then from (8.6)

$$\bar{B}_2 C_{r_2-n_2} \subseteq \Theta_2 \Theta' \bar{A}_2 \bar{A}_1 C_{n_1} \tag{8.9}$$

Together, (8.7) and (8.9) imply

$$[\bar{A}_2 \bar{B}_1, \bar{B}_2]C_{r_1-n_1 + r_2-n_2} \subseteq (\Theta_1 + \Theta_2 \Theta')\bar{A}_2 \bar{A}_1 C_{n_1} \tag{8.10}$$

But the stability number for backward analysis for Algorithm 12 is

$$\Theta_{12} = \text{glb}\{\Theta \geq 0: \bar{B}_{12} C_{r_1-n_1 + r_2-n_2} \subseteq \Theta \bar{A}_{12} C_{n_1}\} \tag{8.11}$$

and hence,

$$\Theta_{12} \leq \Theta_1 + \Theta_2 \Theta' \tag{8.12}$$

While (8.12) shows how the stability number for backward analysis for Algorithm 12 may be related to those of Algorithms 1 and 2, the evaluation of this bound is complicated by the necessity of computing Θ' in (8.8) by one of the exhaustive methods. Nevertheless, the amount of computation will certainly be much less than that required to compute Θ_{12} by exhaustive methods.

Finally, the stability number for B-analysis for Algorithm 12 can be bounded in terms of those for Algorithms 1 and 2. Let the stability numbers for B-analysis for Algorithms 1 and 2 be denoted by

$$\bar{\Theta}_1 = \text{glb}\{\Theta \geq 0: B_1 C_{r_1-n_1} \overset{\cdot\cdot}{\subseteq} \Theta(C_{m_1} + \bar{A}_1 C_{n_1})\} \tag{8.13}$$

and

$$\bar{\Theta}_2 = \text{glb}\{\Theta \geq 0: \bar{B}_2 C_{r_2-n_2} \subseteq \Theta(C_{m_2} + \bar{A}_2 C_{n_2})\} \tag{8.14}$$

respectively. From (8.13) it follows that

$$\bar{A}_2 \bar{B}_1 C_{r_1-n_1} \subseteq \bar{\Theta}_1(\bar{A}_2 C_{m_1} + \bar{A}_2 \bar{A}_1 C_{n_1}) \tag{8.15}$$

Combining (8.14) and (8.15),

$$\bar{A}_2 \bar{B}_1 C_{r_1-n_1} + \bar{B}_2 C_{r_2-n_2} \subseteq \bar{\Theta}_2 C_{m_2} + (\bar{\Theta}_1 + \bar{\Theta}_2)\bar{A}_2 C_{n_2} + \bar{\Theta}_1 \bar{A}_2 \bar{A}_1 C_{n_1} \tag{8.16}$$

Define Θ'' by

$$\Theta'' = \text{glb}\{\Theta \geq 0: \bar{A}_2 C_{n_2} \subseteq \Theta C_{m_2}\}$$

Clearly, $\Theta'' = \|\bar{A}_2\|_\infty$. Then (8.16) becomes

$$[\bar{A}_2 \bar{B}_1, \bar{B}_2]C_{r_1-n_1 + r_2-n_2} \subseteq (\bar{\Theta}_2 + (\bar{\Theta}_1 + \bar{\Theta}_2)\Theta'')C_{m_2} + \bar{\Theta}_1 \bar{A}_2 \bar{A}_1 C_{n_1} \tag{8.17}$$

But the stability number for B-analysis for Algorithm 12 is

$$\bar{\Theta}_{12} = \text{glb}\{\Theta \geq 0: \bar{B}_{12} C_{r_1-n_1 + r_2-n_2} \subseteq \Theta(C_{m_2} + \bar{A}_{12} C_{n_1})\}$$

and hence

$$\bar{\Theta}_{12} \leq \max((\bar{\Theta}_2 + (\bar{\Theta}_1 + \bar{\Theta}_2)\Theta''), \bar{\Theta}_1)$$

9. Algorithm Improvement

From the analysis of an algorithm it is possible to detect those operations which have contributed the most to the poor performance of the algorithm. Knowing which operations are at fault may provide information as to where the algorithm might be changed to improve it. This information is data dependent and enables only a limited improvement for the algorithm evaluated for a specific data. Mathematical identities are used to replace the offending operations.

One goal in algorithm improvement might be to reduce the forward error bounds, $|[\rho\eta_i]_T|$, $1 \leq i \leq m$. Since $\|\overline{B}\|_\infty = \max|[\rho\eta_i]_T|$, reducing the forward bounds may decrease the norm of $\overline{B}$. By decreasing the norm of $\overline{B}$ the polytope for the algorithm, $\overline{B}\, C_{r-n}$, also shrinks so that the backward or B-analysis may determine a smaller stability number, Θ.

Recall (5.6),

$$|[\rho\eta_i]_T| \leq \sum_{j=n+1}^{r} |\rho\eta_i/\rho\xi_j| \tag{9.1}$$

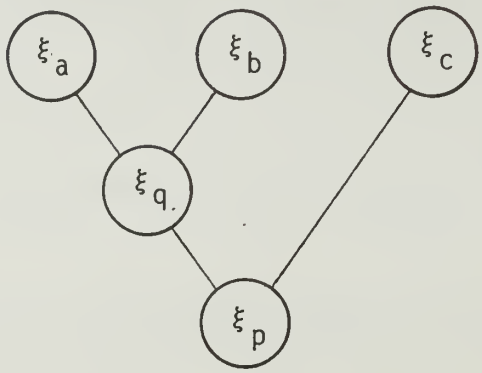
Let I denote a set of indices of nodes in a sub-graph of the algorithm. Then (9.1) may be expressed as

$$|[\rho\eta_i]_T| \leq \sum_{j\in I} |\rho\eta_i/\rho\xi_j| + \sum_{j\notin I} |\rho\eta_i/\rho\xi_j| \tag{9.2}$$

Let I' denote a set of indices of nodes for a sub-graph which is mathematically equivalent to that specified by I. Then by substituting the new sub-graph, (9.2) becomes

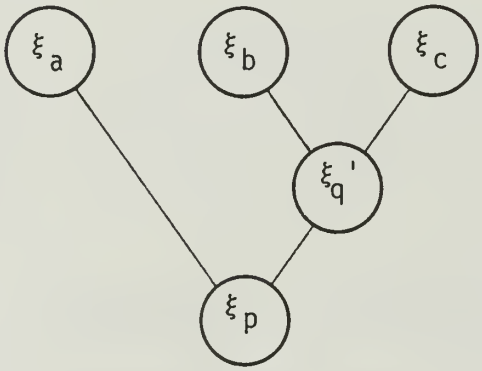
$$|[\rho\eta_i]_T| \leq \sum_{j\in I'} |\rho\eta_i/\rho\xi_j| + \sum_{j\notin I} |\rho\eta_i/\rho\xi_j| \tag{9.3}$$

If the substitution leads to a decrease in the value of the first sum, then the bound has also been correspondingly decreased. Consider a simple case in which I and I' consist of only one node, say ξ_q.



(9.4)

The sub-graph, ξ_q, and surrounding nodes portray the statements $\xi_q = \xi_a * \xi_b$, and $\xi_p = \xi_q * \xi_c$, where * is any one of (+, -, ×, ÷). If the statements $\xi_{q'} = \xi_b * \xi_c$, and $\xi_p = \xi_a * \xi_{q'}$, corresponding to the graph



(9.5)

are equivalent to (9.4), i.e., both mathematically produce the same value for ξ_p, then (9.5) may be substituted for (9.4). Also, this change was possible only if ξ_q was an operand for ξ_p and no other node. Because of this, all the quantities $\rho\eta_i/\rho\xi_j$ remain unchanged, except $\rho\eta_i/\rho\xi_q$ becomes $\rho\eta_i/\rho\xi_{q'}$. Since

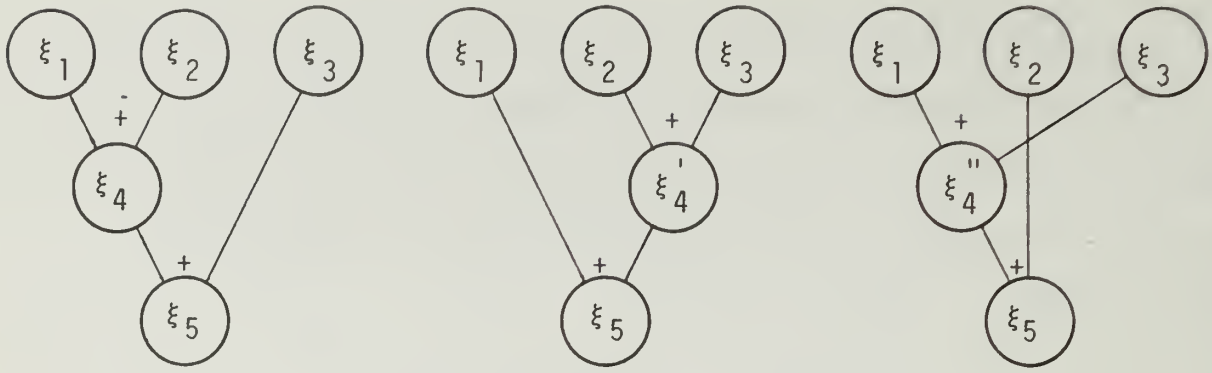
$$\rho\eta_i/\rho\xi_q = (\rho\eta_i/\rho\xi_p)(\rho\xi_p/\rho\xi_q)$$

and (9.6)

$$\rho\eta_i/\rho\xi_{q'} = (\rho\eta_i/\rho\xi_p)(\rho\xi_p/\rho\xi_{q'})$$

the change in $\rho\eta_i/\rho\xi_q$ and hence, in the bounds, $|[\rho\eta_i]_T|$, is governed by the change in the arc weight from $\rho\xi_p/\rho\xi_q$ to $\rho\xi_p/\rho\xi_{q'}$. This fact supports the idea of investigating those sub-graphs which contain large arc weights, since it is there that the greatest improvement can be made. However, each time an arc weight is reduced there is some nearby arc weight which is increased. This phenomenon is a result of the fact that the path product between unchanged nodes must remain the same.

Certainly, sub-graphs corresponding to algebraic identities are mathematically equivalent. The simplest such identities are the associated and distributive laws. These laws correspond to five node graphs similar to (9.4) and (9.5). For the associative law of addition, the following three graphs are interchangeable:



and correspond to the identities

$$(\xi_1 + \xi_2) + \xi_3 \quad = \quad \xi_1 + (\xi_2 + \xi_3) \quad = \quad (\xi_1 + \xi_3) + \xi_2$$

If the first graph were a sub-graph of an algorithm, then successive replacements of it by the other two would replace the term $|\rho\eta_i/\rho\xi_4|$ in (9.1) by $|\rho\eta_i/\rho\xi_4'|$ and then $|\rho\eta_i/\rho\xi_4''|$. By (9.6), the change in $|[\rho\eta_i]_T|$ would be determined by the values of $\rho\xi_5/\rho\xi_4$, $\rho\xi_5/\rho\xi_4'$, and $\rho\xi_5/\rho\xi_4''$, an arc weight in each graph. Clearly, the substitution to be made should choose the graph with the smallest absolute arc weight. The reduction in the arc weight between ξ_4 and ξ_5 will be compensated for by an increase in an arc weight between the new ξ_4 and one of its operands. Thus, the overall effect is to move large arc weights toward the inputs, i.e., any operations involving cancellation should be done early in the computation. This suggests that the graph be investigated from the bottom up, so that fewer and fewer paths contain the arcs with large weights.

Since the weight of an incoming arc to an additive node is a ratio of the operand to result, the graph with the smallest absolute arc weight will be the one which computes the smallest ξ_4 in absolute value.

This fact may be generalized to show the best way to compute $\eta = \sum_{i=1}^{n} \xi_i$.

Any algorithm to compute η may be specified by the sequence of partial sums ξ_{n+i}, $1 \leq i \leq n-1$, where each ξ_{n+i} is the sum of two inputs, one input and a previous partial sum, or two previous partial sums. Since

$$|[\rho\eta]_T| \leq \sum_{i=1}^{n-1} |\rho\eta/\rho\xi_{n+i}|$$

and

$$\rho\eta/\rho\xi_{n+i} = \xi_{n+i}/\eta$$

the forward error bound can be minimized by choosing the partial sums so as to minimize

$$\sum_{i=1}^{n-1} |\xi_{n+i}|$$

For example, if $\xi_1 = 1$, $\xi_2 = 1.25$, $\xi_3 = 1.5$, and $\xi_4 = 1.75$, then the algorithm

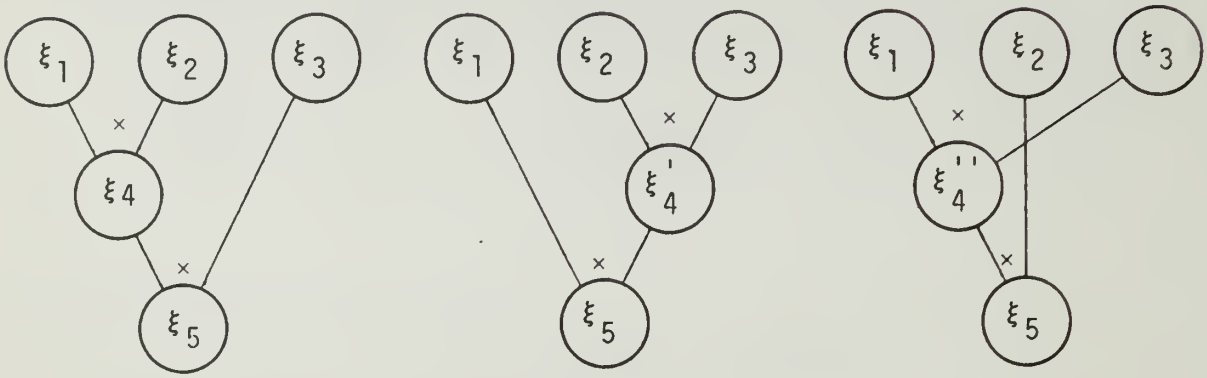
$$\begin{aligned} \xi_5 &= \xi_1 + \xi_2 = 2.25 \\ \xi_6 &= \xi_3 + \xi_4 = 3.25 \\ \eta = \xi_7 &= \xi_5 + \xi_6 = 5.50 \end{aligned}$$

has a forward error bound of 2, while the algorithm which adds the numbers in ascending order

$$\begin{aligned} \xi_5 &= \xi_1 + \xi_2 = 2.25 \\ \xi_6 &= \xi_5 + \xi_3 = 3.75 \\ \eta = \xi_7 &= \xi_6 + \xi_4 = 5.50 \end{aligned}$$

has a forward error bound of $46/22 > 2$.

For the associative law of multiplication, the following three graphs are interchangeable.

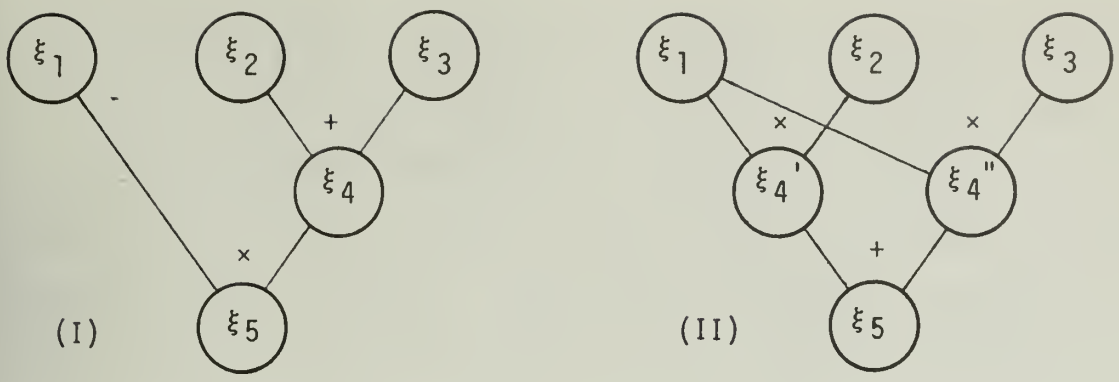


and correspond to the identities

$$(\xi_1 \times \xi_2) \times \xi_3 \quad = \quad \xi_1 \times (\xi_2 \times \xi_3) \quad = \quad (\xi_1 \times \xi_3) \times \xi_2$$

However, since the weights of incoming arcs to multiplicative nodes are 1, no immediate improvement can be made by interchanging these graphs. Such interchanges may however prevent overflow or underflow at the intermediate node ξ_4.

For the distributive law of multiplication over addition, the following two graphs are interchangeable.



and correspond to the identities

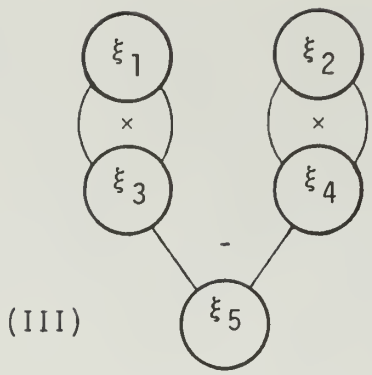
$$\xi_1 \times (\xi_2 + \xi_3) \qquad = \qquad (\xi_1 \times \xi_2) + (\xi_1 \times \xi_3)$$

This law is similar to the other distributive laws involving $\xi_1 \times (\xi_2 - \xi_2)$, $(\xi_2 + \xi_3)/\xi_1$, and $(\xi_2 - \xi_3)/\xi_1$, and what follows is applicable to these variants. The choice between the two graphs involves comparing the values of $|\rho\xi_5/\rho\xi_4|$ and $|\rho\xi_5/\rho\xi_4'| + |\rho\xi_5/\rho\xi_4''|$. While the first is always equal to 1, the second may be much larger. Since the sum of weights of incoming arcs to additive nodes equals 1, i.e., $\rho\xi_5/\rho\xi_4' + \rho\xi_5/\rho\xi_4'' = 1$, it follows that $|\rho\xi_5/\rho\xi_4'| + |\rho\xi_5/\rho\xi_4''| \geq 1$. Thus, the graph (II) can never be better than graph (I). As with the associative law of addition, the trend is to move larger arc weights toward the inputs.

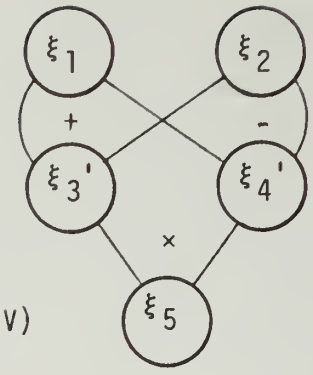
Finally, consider the identity

$$\xi_1^2 - \xi_2^2 = (\xi_1 + \xi_2)(\xi_1 - \xi_2)$$

and associated graphs



(III)



(IV)

The choice here depends on the values of $|\rho\xi_5/\rho\xi_3| + |\rho\xi_5/\rho\xi_4|$ and $|\rho\xi_5/\rho\xi_3'| + |\rho\xi_5/\rho\xi_4'|$. The latter always has value 2, while the first may have any value greater than 1. Conservatively, the choice should be graph (IV). Again, the trend is to have any operations involving cancellation done early in the computation.

The previous discussion has shown that offending operations may be detected by the presence of large values of the arc weights leading from their nodes. Through the use of mathematical identities, improvements by graph substitution can be made which reduce the bounds for forward analysis and hence other analyses as well. The underlying goal of these improvements is to force large arc weights to the arcs leading from input nodes where they will not contribute to the bound in (9.1). The identities described involve very small graphs and hence are limited in their effectiveness. Additionally, a graph substitution often enables another substitution to be made in nearby nodes. The inability to know when to apply the associative law of multiplication may inhibit some later substitution, and suggests that greater improvement can be made by considering larger equivalent graphs. However, this technique is limited by the fact that the computational graph

does not contain all the information about the algorithm or the problem it solves. For example, the solution x of Ax=b satisfies each row equation, and the columns, q_i, of Q in A=QR satisfy $q_i^t q_j = 0$, $i \neq j$. Such information is not available from the inspection of graph nodes or algebraic identities. However, it is this type of information which enables one to transform Cramer's rule to Gaussian elimination, and Gram-Schmidt to modified Gram-Schmidt. Algorithm improvement which could utilize this additional information would have great consequences.

10. Conclusion

Methods have been presented for the automatic error analysis of numerical algorithms. These methods have been developed using Bauer's approach to relative error propagation. Basic to this approach are the concepts of local and total relative errors, and the various systems of equations which relate them. An algorithm is presented which makes use of the equivalence of these systems in order to efficiently compute the equations employed by the error analyses. A modification of the algorithm is also shown to efficiently compute the system of equations used by Miller in his approach to error analysis.

Three error analyses are presented. A forward analysis, which finds a relative error bound for the computed solution, is shown to be easily calculated from the coefficients of the system relating local and total relative errors. This computational ease is contrary to a remark on manual analysis by Wilkinson [19, p. 3] which states that, "In practice, we have found that the backward analysis is often much the simpler." Computationally, the backward analysis, and the B-analysis as well, have proven to be much more difficult than the forward analysis. These analyses, which evaluate the stability of the algorithm, are identified as minimax problems, and as such required tremendous work for their solution, and could only be applied to small problems. A simplification of these analyses significantly reduced the work required, but nonetheless, the work remained exponential or combinatorial in form. An approximation method, which is linear in comparison to the exact methods, is proposed and experimental results indicate that the approximation is very good. The analyses presented are all worst case analyses. This is caused by the use of the ∞-norm in the measurement of

relative errors. The results of the analyses may, therefore, be pessimistic.

Various spin-offs of the main theme are investigated. First, different criteria are presented for the comparison of algorithms. The choice between two algorithms depends on the criteria used. Second, error analyses are derived for composite algorithms, which are made up of smaller sub-algorithms. Bounds for forward, backward, and B-analysis of the composite algorithm are given in terms of corresponding analyses of the sub-algorithms. Finally, by using a forward analysis, limited algorithm improvement is obtained by the location of sensitive operations and the substitution of mathematically equivalent expressions.

Several problems remain unsolved and point the way for future research. The most nagging is the requirement that the value of each node in the computational graph be non-zero. This is a result of the use of relative error in the description of the numerical effects of floating point arithmetic. Arc weights for additive nodes with zero value are infinite. Possible solutions include the replacement of zero by a non-zero quantity which computationally acts as zero, or the assignment of huge arc weights for those defined as infinity. Alternatively, the algorithm may be modified, as in Miller [9], to remove some of the offending nodes altogether.

The simplification of polytope representation in section 6.2 is performed by taking inner products of the columns of $\overline{B}$ and $\dot{A}$ to find those columns which may be condensed. This technique is computationally expensive. An alternative method may perform a condensation of the nodes in the computational graph, before the matrices are formed. A condensed node would be made up of nodes to which the outputs have similar sensitivities. The

sets, S_j, defined in theorem 6.3 may provide a clue as to which nodes may be condensed.

Much remains to be done in the area of algorithm improvement. Ideally, methods for algorithm improvement should be data independent. This, however, may not be possible since data can usually be constructed so that any algorithm performs poorly. The best way to solve a problem may depend directly on the data, and the best algorithm may be different for various chosen data. An alternative approach may seek to find the set of data for which a given algorithm performs satisfactorily. To be effective, either approach must incorporate properties of the problem and algorithm not found in the computational graph.

Naturally, an algorithm may be improved in other ways. In particular, parallel processing imposes other desirable criteria. Bounds on processors, execution time, and storage required become equally important considerations. These quantities are reflected in the height and breadth of the computational graph. The error analyses described herein can serve to evaluate the tradeoff between numerical stability and other criteria.

Bibliography

[1] Ascher, U., "Linear Programming Algorithms for the Chebyshev Solution to a System of Consistent Linear Equations," SIAM J.N.A., Vol. 10, No. 2, April 1968.

[2] Barrodale, I. and F. D. K. Roberts, "An Improved Algorithm for Discrete ℓ_1 Linear Approximation," SIAM J.N.A., Vol. 10, No. 5, October 1973.

[3] Barrodale, I. and F. D. K. Roberts, "Solution of an Overdetermined System of Equations in the ℓ_1 Norm," Algorithm 478, Communications of the ACM, Vol. 17, No. 6, June 1974.

[4] Bauer, F. L., "Computational Graphs and Rounding Error," SIAM J.N.A., Vol. 11, No. 1, March 1974.

[5] Householder, A., The Theory of Matrices in Numerical Analysis, Blaisdell, 1964.

[6] Larson, J. and A. Sameh, "Efficient Calculation of the Effects of Roundoff Errors," to appear in ACM TOMS.

[7] Linnainmaa, S., "Taylor Expansion of the Accumulated Rounding Error," BIT, Vol. 16, No. 2, 1976.

[8] Miller, W., "Roundoff Analysis by Direct Comparison of Two Algorithms," SIAM J.N.A., Vol. 13, No. 3, June 1976.

[9] Miller, W., "Roundoff Analysis and Sparse Data," Numerische Mathematik, Vol. 29, No. 1, 1977.

[10] Miller, W., "Software for Roundoff Analysis," ACM TOMS, Vol. 1, No. 2, June 1975.

[11] Miller, W., "Computer Search for Numerical Instability," Journal of the ACM, Vol. 22, No. 4, October 1975.

[12] Miller W. and D. Spooner, "Software for Roundoff Analysis, II," to appear in ACM TOMS.

[13] Nijenhuis, A. and H. S. Wilf, Combinatorial Algorithms, Academic Press, 1975.

[14] Panzer, K., "Gutartigkeit von Rechenprozessen," Technische Universität München, Fachbereich Mathematik, Interner Bericht, September 1975.

[15] Rabinowitz, P., "Applications of Linear Programming to Numerical Analysis," SIAM Review, Vol. 10, No. 2, April 1968.

[16] Sameh, A. and R. Brent, "Solving Triangular Systems on a Parallel Computer," SIAM J.N.A., Vol. 14, No. 6, December 1977.

[17] Sterbenz, P. H., Floating Point Computation, Prentice Hall, 1974.

[18] Stewart, G. W., Introduction to Matrix Computations, Academic Press, 1973.

[19] Wilkinson, J. H., Rounding Errors in Algebraic Processes, Prentice Hall, 1963.

Appendix

Description of algorithms tested.

BACK4, BACK8: an algorithm for forward substitution on a unit lower triangular matrices of dimension 4 and 8. For data used see PARA4.

SEMI4, SEMI8: an algorithm for solving unit lower triangular systems $Ax=b$ of dimension 4 and 8 according to the following scheme:

Let $A = L_1^{-1}, L_2^{-1} \ldots L_{n-1}^{-1}$, where $L_k = (\lambda_{ij})$ is the identity except in the k^{th} column where $\lambda_{ij} = -a_{ij}$. Then $x = (L_3(L_2(L_1\ b)))$ for dimension 4 and similarly for dimension 8. For data used see PARA4.

PARA4, PARA8 [16]: similar to SEMI4, SEMI8, except log product is used, i.e., $x = (L_3\ L_2)(L_1\ b)$ for dimension 4 and similarly for dimension 8. Data used for dimension 4 was

$$A = \begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ -3 & -5 & 1 & \\ -4 & -6 & -7 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

For dimension 8 the data was

$$A = \begin{bmatrix} 1 & & & & & & & \\ -1 & 1 & & & & & & \\ -2 & -8 & 1 & & & & & \\ -3 & -9 & -14 & 1 & & & & \\ -4 & -10 & -15 & -19 & 1 & & & \\ -5 & -11 & -16 & -20 & -23 & 1 & & \\ -6 & -12 & -17 & -21 & -24 & -26 & 1 & \\ -7 & -13 & -18 & -22 & -25 & -27 & -28 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

LTBAND: an algorithm by Sameh, Brent [16] for the parallel solution of banded lower triangular matrices. The data used was

$$A = \begin{bmatrix} -1 & & & & & & & \\ 4 & -1 & & & & & & \\ -1 & 4 & -1 & & & & & \\ & -1 & 4 & -1 & & & & \\ & & -1 & 4 & -1 & & & \\ & & & -1 & 4 & -1 & & \\ & & & & -1 & 4 & -1 & \\ & & & & & -1 & 4 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

MATMUL: an algorithm for multiplying two matices of dimension 4. The data was

$$A = \begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 5 & 9 & 13 \\ -2 & -6 & -10 & -14 \\ 3 & 7 & 11 & 15 \\ -4 & -8 & -12 & -16 \end{bmatrix}$$

RINVERSE: an algorithm for inverting an upper triangular matrix.

The data [12] was

$$A = \begin{bmatrix} 0.0004 & 4.5692 & 4.8228 & 3.1819 \\ & 3.9494 & 4.1681 & 3.7357 \\ & & 0.004 & -5.0996 \\ & & & 4.4464 \end{bmatrix}$$

CHOL: the Cholesky method applied to the data [12]

$$A = \begin{bmatrix} 3 & \text{(symmetric)} & & \\ 1 & 4 & & \\ 1 & 1 & 5 & \\ 1 & 1 & 1 & 6 \end{bmatrix}, \quad b = \begin{bmatrix} 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}$$

G.E.: Gaussian elimination without pivoting with data [12] as in CHOL.

C.G.: The method of conjugate gradients applied to the data [12]

$$A = \begin{bmatrix} 3 & \text{(sym)} & \\ 1 & 4 & \\ 1 & 1 & 5 \end{bmatrix}, \quad b = \begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix}$$

G.J.: Gauss-Jordan elimination with data as in CHOL.

G.S.: The Gram-Schmidt orthogonalization algorithm applied to the three 4-vectors

$$v_1 = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}, \quad v_2 = \begin{bmatrix} -6 \\ -6 \\ 14 \\ 14 \end{bmatrix}, \quad v_3 = \begin{bmatrix} -13 \\ 7 \\ -5 \\ 15 \end{bmatrix}$$

M.G.S.: The modified Gram-Schmidt algorithm with data as in G.S.

Vita

The author, John L. Larson, was born February 6, 1949, in Champaign, Illinois. He received B.S., M.S., and Ph.D. degrees from the University of Illinois at Urbana-Champaign in the years 1971, 1974, and 1978, respectively. During his graduate education, he was employed as a teaching and research assistant with the Department of Computer Science.

The titles of his publications include: "Efficient Calculation of the Effects of Roundoff Errors," (with Ahmed H. Sameh) and "Automatic Error Analysis for Serial and Parallel Algorithms."

He is a member of the Association for Computing Machinery and the Society of Industrial and Applied Mathematicians.

ILIOGRAPHIC DATA ET	1. Report No. UIUCDCS-R-78-916	2.	3. Recipient's Accession No.
itle and Subtitle			5. Report Date April, 1978
Methods for Automatic Error Analysis of Numerical Algorithms			6.
uthor(s) John Leonard Larson			8. Performing Organization Rept. No. UIUCDCS-R-78-916
erforming Organization Name and Address			10. Project/Task/Work Unit No.
University of Illinois at Urbana-Champaign Department of Computer Science Urbana, Illinois 61801			11. Contract/Grant No. US NSF MCS75-21758
Sponsoring Organization Name and Address			13. Type of Report & Period Covered Doctoral Dissertation
National Science Foundation Washington, D. C.			14.

Supplementary Notes

Abstracts

sing Bauer's approach to relative error propagation, methods are presented for erforming various error analyses on numerical algorithms. A technique is developed hich generates a system of equations employed by the error analyses, and requires an rder of magnitude less time and storage than present techniques. Accurate analyses re attempted by working in the ∞-norm, and require the solution of minimax problems. heuristic approximation method is also described. These methods are compared with he 2-norm approximation methods of Miller.

he error analyses provide alternative criteria by which algorithms that solve the ame problem may be compared. Additionally, the analysis of a composite algorithm, hich is made up of concatenated sub-algorithms is given in terms of analyses done on ts parts. Finally, by using a forward analysis, limited algorithm improvement is btained by the location of sensitive operations, and the substitution of mathematically equivalent expressions.

Key Words and Document Analysis. 17a. Descriptors

utomatic error analysis
umerical stability
inear algebra

. Identifiers/Open-Ended Terms

. COSATI Field/Group

Availability Statement	19. Security Class (This Report)	21. No. of Pages
ELEASE UNLIMITED	UNCLASSIFIED	106
	20. Security Class (This Page) UNCLASSIFIED	22. Price

RM NTIS-35 (10-70)

USCOMM-DC 40329-P71

JUN 2 0 1978

BOUND BY
HNM